VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2019 Bc. Daniel Michalík

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV MIKROELEKTRONIKY
DEPARTMENT OF MICROELECTRONICS

ŘÍZENÍ A KONSTRUKCE VÍCEÚČELOVÉHO OBRABĚCÍHO STROJE
CONSTRUCTION AND CONTROL OF MULTIPURPOSE MILLING MACHINE

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Daniel Michalík**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Josef Eliáš**
SUPERVISOR

BRNO 2019

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Mikroelektronika**
Ústav mikroelektroniky

Student: Bc. Daniel Michalík ***ID:*** 173704
Ročník: 2 ***Akademický rok:*** 2018/19

NÁZEV TÉMATU:

Řízení a konstrukce víceúčelového obraběcího stroje

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je návrh a testování řízení a grafického rozhraní obráběcího stroje vybrané konstrukce.
Student v první části vybere vhodnou konstrukci obráběcíhoho stroje a následně navrhne vhodný způsob řízení tak, aby bylo možné využít více způsobů pro výrobu mechanických dílů.

DOPORUČENÁ LITERATURA:

Podle pokynů vedoucího práce

Termín zadání: 4.2.2019 ***Termín odevzdání:*** 21.5.2019

Vedoucí práce: Ing. Josef Eliáš
Konzultant:

doc. Ing. Lukáš Fujcik, Ph.D.
předseda oborové rady

ABSTRAKT

Práce se zabývá návrhem a realizací víceúčelového obráběcího stroje pro účely výroby prototypů drobných dílu z měkkých materiálů a DPS. Práce obsahuje návrh mechanické konstrukce, jednotlivých částí řízení stroje a ovládacího rozhraní.

KLÍČOVÁ SLOVA

CNC, obrábění, ZYNQ, FPGA, ARM, RTOS, vřeteno, frézka

ABSTRACT

This semestral work deals with the design and realization of multipurpose milling machine for production of prototype small components made from soft materials and PCBs. Thesis contains design of mechanical construction, individual parts of driving machine and graphic user interface.ě

KEYWORDS

CNC, milling, ZYNQ, FPGA, ARM, RTOS, spindle, router

MICHALÍK, Daniel. *Řízení a konstrukce víceúčelového obraběcího stroje*. Brno, Rok, 54 s. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav mikroelektroniky. Vedoucí práce: Ing. Josef Eliáš

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Řízení a konstrukce víceúčelového obraběcího stroje“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu Ing. Josefu Eliášovi za cenné rady k návrhu, konzultace detailů jednotlivých částí a trpělivost. Dále SW teamu firmy ELEDUS s.r.o. za rady během návrhu a Lukášovi Bezdíčkovi za pomoc při návrhu konstrukce.

Brno ....

podpis autora

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Purkynova 118, CZ-61200 Brno
Czech Republic
http://www.six.feec.vutbr.cz

PODĚKOVÁNÍ

Výzkum popsaný v této diplomové práci byl realizován v laboratořích podpořených z projektu SIX; registrační číslo CZ.1.05/2.1.00/03.0072, operační program Výzkum a vývoj pro inovace.

Brno ....

podpis autora

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

OP Výzkum a vývoj
pro inovace

Obsah

Úvod

Tato práce se věnuje návrhu víceúčelového obráběcího stroje. Cílem je zvolit efektivní řešení dané problematiky zejména po stránkách mechanické konstrukce, řídících částí a hardwarových komponent.

Každou část této problematiky je možné řešit několika způsoby. Jednotlivá řešení budou v teoretické části práce podrobně popsány a zhodnoceny. Následně bylo zvoleno nejvhodnější řešení.

První kapitola popisuje teorii o obráběcích strojích. Zmiňuje zařazení CNC frézky mezi zástupci jednotlivých strojů určených pro třískové obrábění. Popisuje souřadnicový systém stroje, hlavní mechanické komponenty a kritické části návrhu, které mají vliv na výslednou kvalitu stroje.

Druhá kapitola se věnuje teorii o řízení obráběcích strojů. Popisuje význam interpolátoru, strukturu řídícího programu a základní příkazy. Vysvětluje co jsou to G-kódy, CAM a postprocesor. Popisuje průmyslově vyráběná řešení určené pro stroje určené na výrobu prototypů. Je zmíněn systém na čipu Zynq jako jedno z možných řešení.

Třetí kapitola popisuje návrh jednotlivých částí stroje. Začíná mechanickou konstrukcí a vřetenem. Dále porovnává možnosti řešení řídicí části následované výběrem té nejvhodnější, u které je popsán vlastní komunikační protokol, parser G-kódu a návrh konfiguračního souboru pro postprocesor.

Kapitoly 4-6 obsahují popis realizace řízení na úrovni hradlového pole, procesoru a počítače. Jsou zmíněny způsoby vývoje programů, jejich manuální a automatizované testování.

Poslední kapitola shrnuje dosažené výsledky studentské práce včetně fotografií hotového prototypu stroje.

1 Mechanická konstrukce obráběcího stroje

Obráběcí stroj je stroj určený pro třískové obrábění materiálu. Obrábění je proces při němž je obrobek opracováván do požadovaného tvaru za vzniku třísek. Typickými zástupci jsou například soustruhy, frézky, vrtačky a jiné. Stroje lze podle druhu řízení rozdělit na 2 skupiny: ruční a počítačem řízené. Ruční obráběcí stroje ovládá obsluha přímo, počítačem řízené stroje obsluha nastaví a jejich chod dále řídí počítač pomocí motorů, serv nebo jiných elektromechanických částí. Tato práce se zabývá návrhem a řízením automatického výceúčelového zařízení nejen pro obrábění. V následujících kapitolách budou popsány jednotlivé části stroje a jejich varianty. Na konci kapitoly je popsán výčet možností a výběr vhodné konstrukce pro víceúčelový stroj.

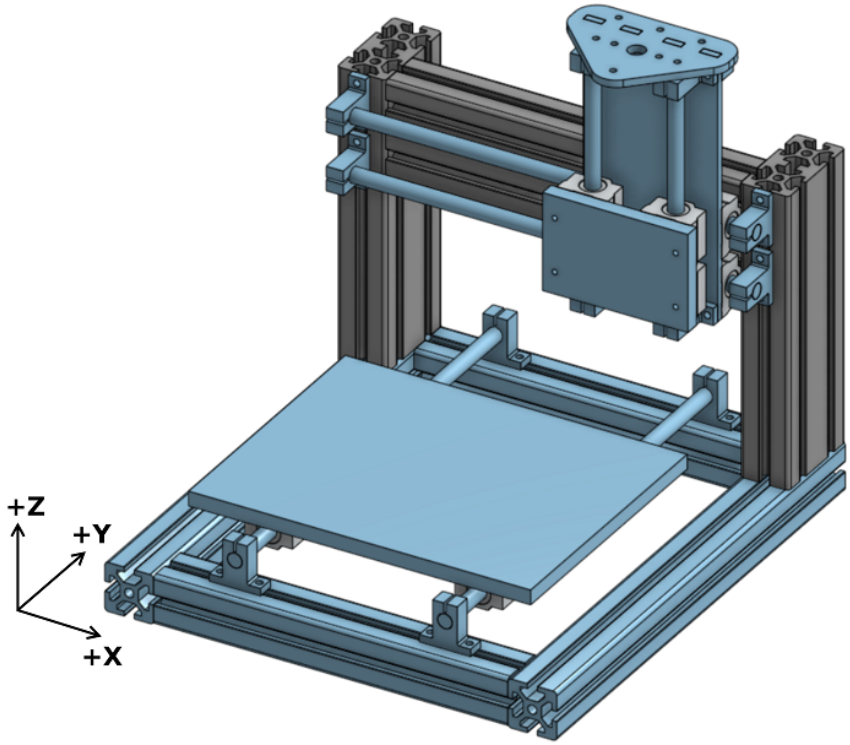
1.1 CNC frézka

Pod názvem CNC frézka se skrývá nejčastěji 3 a více-osý stroj určený pro obrábění. Existují verze dostupné pro veřejnost, tzv. hobby frézky a profesionální stroje pro obrábění tvrdších materiálů. Příklad hobby frézky je na obrázku 1.1. Tato práce je zaměřena na hobby frézku určenou pro obrábění měkkých materiálů a prototypovou výrobu. Pro výrobu DPS je možné pracovní nástroj nahradit například dávkovačem pro nanášení pájecí pasty nebo vakuovou pinzetou pro přesné osazení SMD součástek.

Obr. 1.1: Příklad hobby frézky [1]

1.2 Souřadnicový systém stroje

Definici souřadnicového systému stroje lze nalézt v normě ČSN ISO 841 [2]. Dle této normy se jedná o systém ke stanovení souřadnic nástroje nebo obrobku v pracovním prostoru stroje. V rámci CNC frézky se udává souřadnice nástroje. Hlavní osy jsou označeny písmeny X, Y a Z. Rotace jednotlivých os jsou značeny písmeny A, B a C. Počátek stroje stanovuje výrobce stroje. Rozvržení jednotlivých os je na obrázku 1.2.



Obr. 1.2: Souřadný systém stroje

1.3 Mechanické komponenty

Profesionální stroje jsou konstruovány s ohledem na jejich využití. Tato konstrukce je určena pro hobby práci s více pracovními nástroji, proto bude její konstrukce vybrána s ohledem na variabilitu systému. Volba jednotlivých součástí závisí hlavně na účelu stroje. Stroj je navrhován pro obrábění měkkých materiálů.

1.3.1 Nosný rám

Nosná část určuje tuhost celého stroje. Hlavním kritériem tuhosti je zvolený materiál (modul pružnosti E) a tvar profilů které tvoří samotnou konstrukci. Běžným konstrukčním materiálem pro nosný rám jsou hliníkové profily, litinové prvky a broušené konstrukce z žuly[3].

1.3.2 Lineární vedení

Pohyb jednotlivých os je veden pomocí lineárního vedení. Existuje několik možných řešení, odlišující se hlavně tuhostí a vůlí v uložení. Dle [4] patří mezi nejpoužívanější:

- Vedení s oběhovými kuličkami - lze dosáhnout nízké vůle a velké životnosti, tuhost určuje materiál, na který jsou kolejnice přišroubovány
- Hlazené tyče - využívají lineární ložiska, je možné je namáhat ve všech směrem kolmých na osu, pevnost je přímo úměrná průměru tyče, z toho vyplývá nevhodnost pro dlouhé vedení
- Podepřené tyče - využívají otevřené ložiska, disponují vyšší tuhostí oproti předchozí variantě, ale nevýhodou je omezené namáhání ve směru od osy, tuhost je dána průřezem profilu

U lineárního vedení je důležitá volba materiálu z důvodu tuhosti a odolnosti proti korozi. Zejména při použití chladících kapalin na bázi vody může docházet ke korozi některých částí.

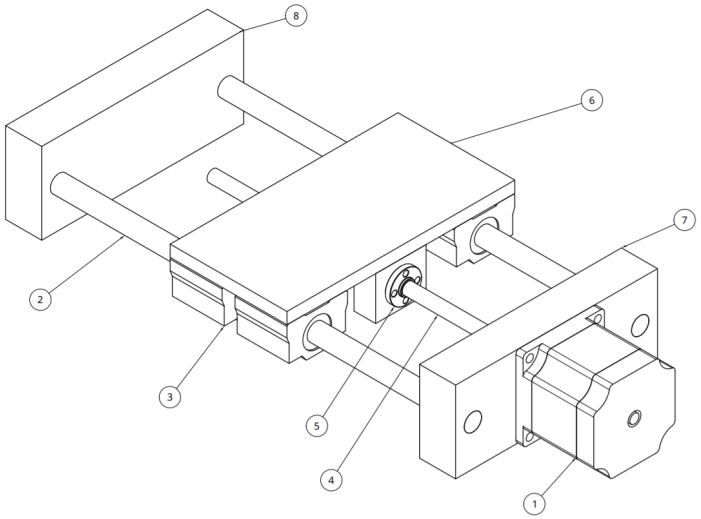
1.3.3 Vřeteno

Vřeteno je rotující část stroje, ve které je upnut nástroj. Konstrukce samotného vřetene má velký vliv na kvalitu obrábění. Je požadována dostatečná tuhost a vysoké otáčky. Otáčky se pohybují v rozmezí jednotek až tisíců otočení za minutu. Pomalejší rychlosti je možné využít například na řezání závitů. Rychlejší otáčky pro obrábění. Maximální rychlost je nutné zohlednit při návrhu. Pro vysokou rychlost byla zvolena vysokorychlostní ložiska s kosoúhlým stykem. Kosoúhlý styk je zvolen z důvodu přesnosti uložení [5]. K její dosažení je nutné ložiska předepnout. Předpětí je síla, která působí mezi kuličkami a kroužky ložiska, která není vyvolaná vnějším namáháním[6]. Je to využíváno u aplikací s požadavkem na vysokou tuhost a přesnost uložení[7].

1.3.4 Pohon os

Pohyb jednotlivých os je možné rozdělit na lineární a rotační. Lineární pohon je typický pro osy X, Y a Z viz. Souřadnicový systém stroje. Rotační osy se používají jako přídavné pro možnost otáčet obrobkem na pracovním stole víceosých strojů [8]. Převod rotačního pohybu motoru na lineární se provádí pomocí šroubu[9], řemene nebo ozubeného hřebenu. Nejčastěji jsou používané šrouby, trapézové nebo kuličkové. U obou typů je nutné vymezit vůli [10]. To se dělá rozdělením matice na 2 poloviny a vymezením vzdálenosti podložkou, pružinou nebo volbou přesněji vyrobených součástí [5].
Na následujícím obrázku 1.3 lze vidět základní části lineární osy:

1. Motor
2. Lineární vedení
3. Lineární ložiska
4. Šroub
5. Matice
6. Pracovní plocha osy
7. Čelo pro fixaci motoru k tyčím
8. Čelo pro fixaci konců tyčí



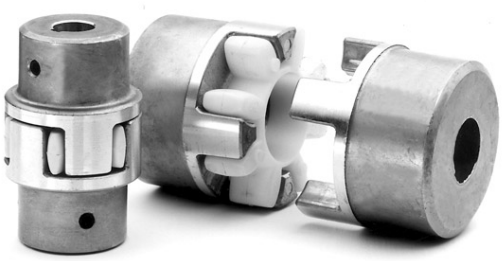
Obr. 1.3: Lineární osa

Motory používané v CNC se dělí na servopohony a krokové motory [11]. Servopohony jsou AC nebo DC motory doplněné zpětnou vazbou o poloze. Řízení je prováděno pomocí PID regulátorů [12]. Při špatném naladění regulátorů může docházet k přejetí cílové polohy. Výhodou je přesnější nastavení pozice. Zpětná vazba

je získávána z absolutního nebo inkrementálního snímače. Nevýhodou inkrementálního snímače je nutnost kalibrace na výchozí bod. Z absolutního snímače lze hodnotu zjistit po restartu stroje bez nutnosti kalibrace. Krokové motory mají pevně danou velikost kroku, která je určena mechanickou konstrukcí rotoru a statoru. Pro dosažení velkého počtu kroků na otáčku se používá několik vzájemně pootočených rotorů. Otáčení zajišťuje driver postupným spínáním cívek v motoru. Podle zapojení cívek se motory dělí na bipolární a unipolární.
Krokové motory lze podle konstrukce rotoru dělit na[13]:

- Krokový motor s aktivním rotorem - s permanentním magnetem
- Krokový motor s pasivním rotorem - využívá magnetický odpor rotoru
- Hybridní krokový motor - kombinace předchozích (permanentní magnet v pouzdře z měkkého materiálu na kterém jsou jednotlivé zuby)

Mezi motor a šroub se vkládá tzv. pružná spojka, která kompenzuje nesouosost osy motoru a šroubu. Spojky jsou tvořeny dvěma přírubami spojenými pružným členem. Spojka typu OLDHAM je na obrázku 1.4.



Obr. 1.4: Pružná spojka typu OLDHAM[14]

2 Řízení obráběcích strojů

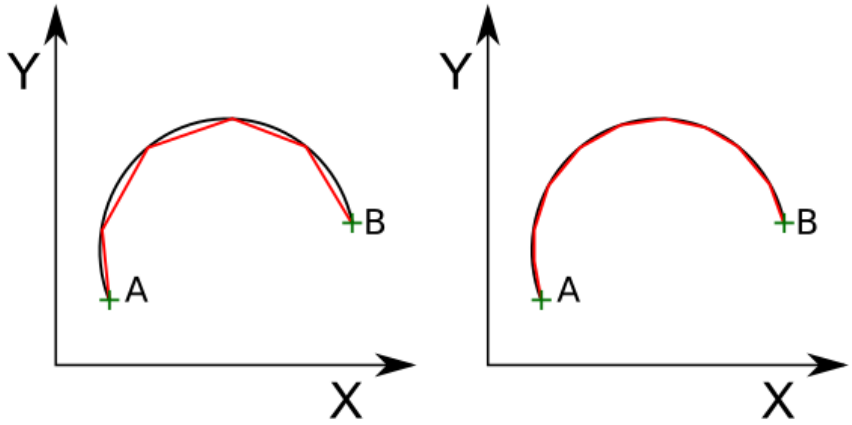
Tato kapitola popisuje způsob, jakým jsou obráběcí stroje programovány a jakým způsobem budou zpracovávat zadané programy. Tuto část lze rozdělit na programování G-kódy (norma DIN 66025) a ostatní.

2.1 Interpolátor

Interpolátor je výpočetní část stroje, která počítá souřadnice jednotlivých bodů trajektorie. Pokud je nutné přesunout nástroj, například po kružnici, je nutné zadat počáteční a koncový bod, poloměr a směr pohybu. Interpolátor vypočítá další body, které jsou nutné pro řízení motorů. Interpolátory lze dělit na:

- lineární
- kruhové
- s vyšším stupněm interpolace (prostorová interpolace)

Číslicové systémy pracují nejčastěji s minimálním krokem. Čím jemnější krok je zvolen, tím je výsledný pohyb přesnější. Příkladem je rozdělení křivek na úsečky, interpolované na minimální krok motoru. Vliv velikosti úseček lze vidět na obrázku 2.1. Při velkém kroku dochází k velké chybě. Při velmi jemném proložení se znásobí počet příkazů, které je nutné do stroje poslat - je potřeba dostatečná propustnost komunikační linky. S tím je potřeba počítat při dalším návrhu stroje.



Obr. 2.1: Vliv velikosti úseček na chybu při prokládání kruhové trajektorie

2.2 NC program

Pohyby stroje jsou řízeny podle NC programu. Jeden řádek programu se skládá z čísla řádku začínajícím písmenem N, slova začínajícím písmenem M nebo G a vstupními parametry. Příklad slova **N0012 G01 X10 Y10** provede posun pracovní rychlostí úhlopříčně 10mm v ose X a v ose Y. Posuny jsou v tomto případě zadány v inkrementálním režimu. Jedná se o relativní režim - zadávají se souřadnice vzhledem k poslední poloze. Druhým možným způsobem je absolutní programování - souřadnice se zadávají vzhledem k nulové souřadnici stroje. Režim se zadává pomocí příkazů G90 a G91.

2.2.1 G kódy

G kódy jsou části programu, které vykonávají pohybové a pomocné funkce [15]. Mezi základní používané funkce patří:

- G00 - lineární interpolace rychloposuvem
- G01 - lineární interpolace pracovní rychlostí
- G02 - kruhová interpolace ve směru hodinových ručiček CW
- G03 - kruhová interpolace proti směru hodinových ručiček CCW
- G17 - nastavení obráběcí roviny XY
- G21 - nastavení jednotek na milimetry
- G90 - absolutní polohování
- G91 - relativní polohování

2.2.2 M kódy

M kódy jsou části programu, které představují pomocné a přípravné funkce [15]. Mezi základní funkce patří:

- M03 - spuštění vřetene, směr otáčení: doprava
- M04 - spuštění vřetene, směr otáčení: doleva
- M05 - zastavení vřetene
- M02, M30 - konec programu

2.3 CAM

G-kód je generován v programu, který se nazývá CAM. Do tohoto programu je nutné vložit model obrobku, zadat parametry jako například průměr, délka a tvar frézy, bezpečná výška pro pohyb nástroje a jiné. Následně je kód nahrán do postprocesoru zajišťující přípravu kódu pro cílový stroj (např. proložení kruhové trajektorie úsečkami, detekce nepodporovaných funkcí, apod.).

2.4 Průmyslově vyráběné řídicí systémy

Základní skupinou jsou rozšiřující moduly pro PC. Nejjednodušší je použití LPT portu přímo pro generování pulzů pro drivery. Tento způsob využívá například otevřená platforma LinuxCNC. Problémem tohoto řešení je hlavně rychlost. Pro správnou funkci je LinuxCNC postaven na real-time rozšíření v linuxu, které je nutné pro přesné řízení.

HW interpolátory jsou periferie k PC osazené procesorem, který interpoluje trajektorii a generuje pulzy pro motory. Procesor je v některých případech doplněn programovatelným hradlovým polem pro dosažení synchronizace na úrovni řízení jednotlivých motorů. Zástupce těchto interpolátorů je například SmoothStepper[16]. Tato deska obsahuje hradlové polem SPARTAN 6.

2.5 ZYNQ

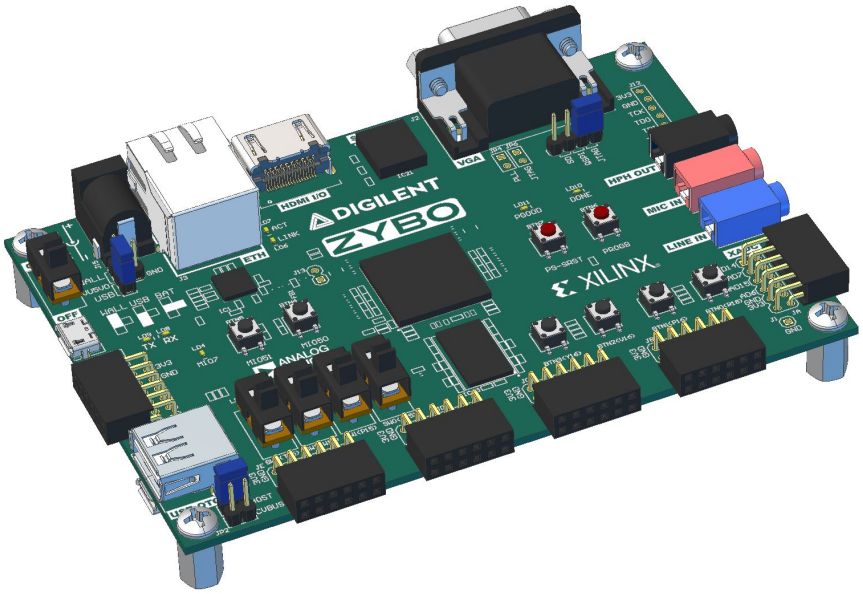
ZYNQ je integrovaný obvod obsahující 2-jádrový procesor ARM A9, hradlové pole a další periferie. Je to tzv. SoC - Systém na čipu. Komunikace mezi ARMem a hradlovým polem probíhá po vnitřní AXI sběrnici. Výhodou je možnost doplnit procesor libovolnou periferií, vytvořenou v rámci hradlového pole a vytvořit tak synchronní akcelerátor pro řízení krokových motorů.

V rámci čipu jsou dvě části PS a PL. Vnitřní blokové schéma je na obrázku 2.3. PS je Processing system - procesor a PL je Programmable logic - hradlové pole. Součástí procesorové části jsou standardní periferie I2C, SPI, UART, CAN, řadič SD karty a ethernetu.

Pro základní seznámení se systémem ZYNQ byl zvolen vývojový kit *Zybo Zynq-7000 ARM/FPGA SoC Trainer Board* viz. 2.2.
Systém disponuje zejména:

- dvoujádrovým procesorem Cortex A9 běžícím na frekvenci 650 MHz
- vnitřní sběrnicí AMBA - Advanced Microcontroller Bus Architecture
- FPGA částí o velikosti 28k logických buňek

Kit lze programovat přímo přes USB port na desce s využitím nástrojů v programu Vivado. Další možností je načtení konfigurace z SD karty. Tato funkce se hodí především pro použití operačního systému.
Deska je osazena IO XC7Z010-1CLG400C. Vnitřní schéma je na obrázku 2.3. Uprostřed lze vidět zmíněný procesor ARM s dvěma jádry doplněnými FPU jednotkami. Procesor je k ostatním periferiím připojen pomocí sběrnice AXI 3. V dolní části lze vidět FPGA, které obsahuje mimo jiné analogově digitální převodník a PCI-express řadič.



Obr. 2.2: Vývojový kit ZYBO board

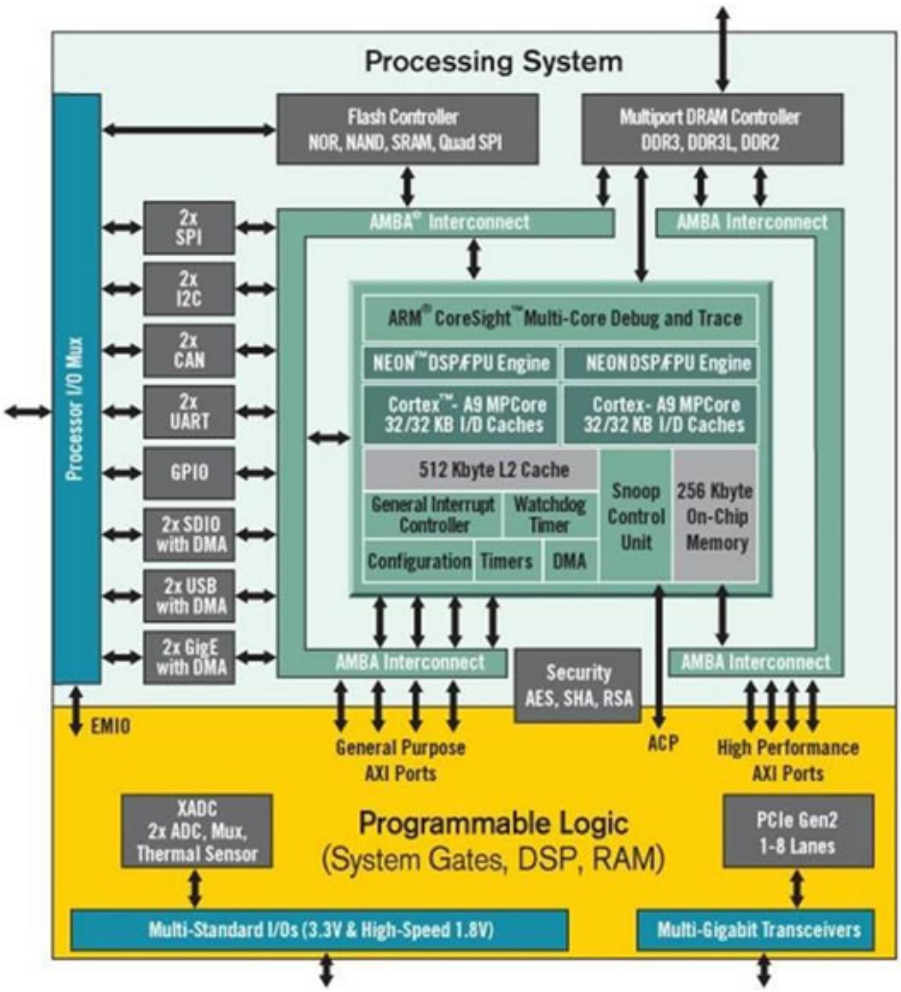
2.5.1 AXI sběrnice

Je to jedna z nejpoužívanějších typů sběrnice pro SoC. Disponuje vysokou propustností a nízkou latencí. Komunikace probíhá v režimu master-slave na pěti kanálech. Master začíná komunikaci, posílá příkaz a slave odpovídá na jeho požadavek. Master může zapisovat nebo číst. Při zápisu je poslána adresa a následně data. Odpovědí je potvrzení. Při čtení je zapsána adresa a odpovědí jsou data.

Existují 3 typy AXI rozhraní:

- Lite - paměťově mapované pro nenáročné aplikace, na jedno nastavení adresy připadá jeden zapsaný rámec dat
- Stream - jednosměrný proud dat, vyniká vysokou propustností
- Memory mapped - paměťově mapované, lze zapisovat v režimu burst (na jedno nastavení adresy lze zapsat až 256 rámců o velikosti 32 bitů)

Jednotlivé bloky na AXI sběrnici lze propojit pomocí bloku propojení - interconnect. Výhodou tohoto bloku je možnost propojit 32-bitovou sběrnici s 64-bitovým. Časování jednotlivých stran může být také libovolné. Lze využít synchronní nebo asynchronní konverzi mezi různými hodinovými doménami. Bloky je možné spojovat do libovolných stromových struktur. Interconnect může propojovat master a slave bloky dle následujících vzorů: N:1, 1:N a N:M. Další funkcí je konverze AXI protokolu mezi AXI 3 a AXI 4.



Obr. 2.3: Systém na čipu Zynq-7000 [17]

2.6 Požadavky na grafické rozhraní

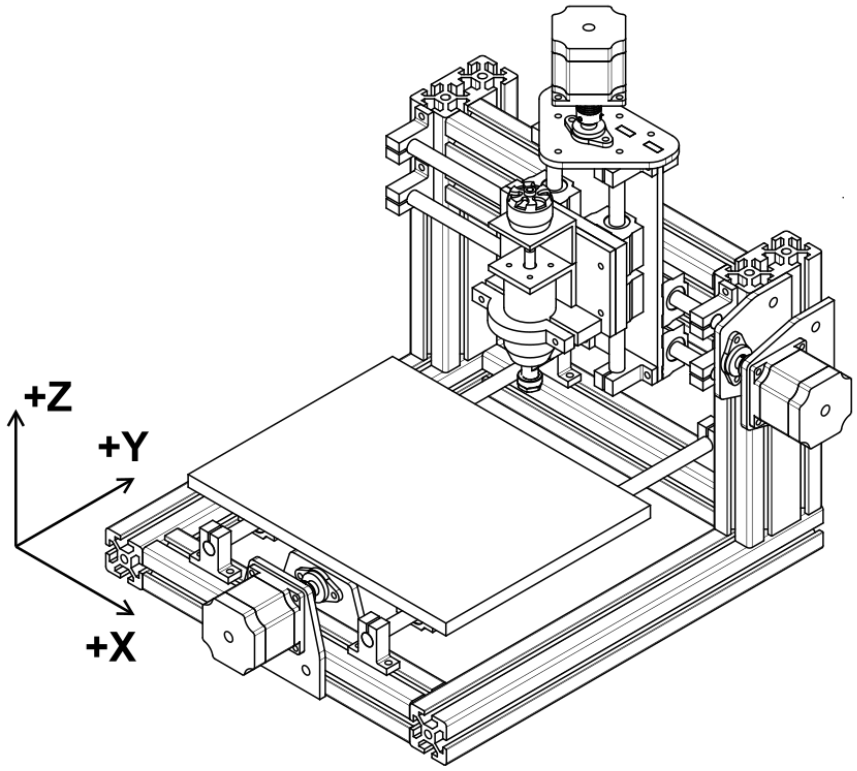
Grafické rozhraní by měl tvořit jednoduchý panel s intuitivním ovládáním. Na strojích je několik důležitých panelů, které by neměly chybět. Základním panelem je textový editor zobrazující právě probíhající program. Možnost editace je výhodou, ale není nutností. Běh programu se spouští, zastavuje a případně krokuje pomocí panelu řízení běhu obsahující tlačítka Start, Stop a Step. Aktuální pozice stroje zobrazuje panel souřadnic s možností nulování jednotlivých os ze SW. Pro nástroj se používá panel řízení nástroje, nastavuje se primárně rychlost a směr otáčení vřetene. Pro spojení se strojem je důležitý panel komunikace.

3 Návrh dílčích částí

3.1 Konstrukce a vřeteno

Pro návrh mechanické konstrukce byl použit cloudový software OnShape[18] pro jeho jednoduchost a dostupnost bez nutnosti instalace. Program disponuje standardními funkcemi pro modelování, vytváření výkresu a export dat pro potřebu výroby.

Návrh celé konstrukce je na obrázku 3.1. Nosná konstrukce je složená z hliníkových profilů. Pro lineární vedení byly zvoleny hlazené tyče průměru 12 mm. Jednotlivé díly z hliníkových plechů jsou tloušťky 6 a 12 mm. V původním návrh bylo uvažováno nahrazení profilů díly z plechu. Toto řešení nebylo použito pro náročné vymezení vůlí stroje.

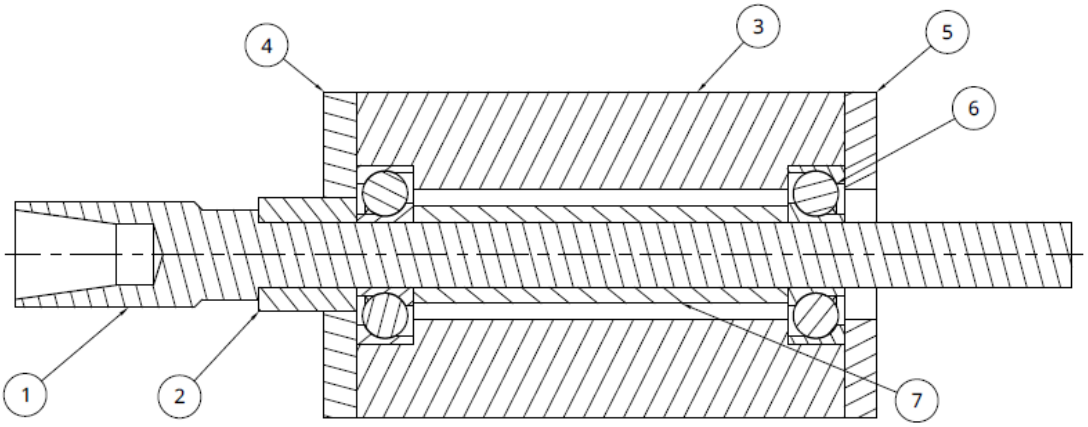


Obr. 3.1: Návrh konstrukce

Počátek stroje byl zvolen do levého předního rohu viz. 3.1. Osa X je rovnoběžná s portálem a je kladná ve směru z leva do prava. Osa Y je kladná ve směru ze předu do zadu. Osa Z je rovnoběžná s osou vřetene a kladná ve směru nahoru. Jedná se o pravotočivý souřadnicový systém.

Vřeteno bylo navrženo s ložisky s kosoúhlým stykem 708AC. Řez lze vidět na obrázku 3.2. Hlavní částí je sklíčidlo ER11 označené číslem 1. Výhodou sklíčidla je

možnost výměny kleštiny v rozsahu 1 - 7 mm. Osa je uložena v ložiscích (6). Vzdálenost hlavy sklíčidla určuje vymezovací člen (2). Ložiska jsou nalisovány v bloku hliníku (3), který zajišťuje jejich souosost. Vymezovací válec (7) vymezuje vůli ložisek a proto musí být velice přesný. Tento díl je nutné vyrobit přesně. V opačném případě je nutné zmenšit délku válce a doplnit jej přesnými vymezovacími podložkami. Čela (4) a (5) tvoří druhý opěrný bod pro ložiska, jsou opřena o jejich vnější kroužek.



Obr. 3.2: Řez vřetenem

Ložiska je nutné předepnout. To se dělá upravením délky vymezovacího válce pro dosažení minimální vůle. Při nízkém předpětí dochází k vibracím a může dojít ke zničení ložisek. Nadměrným předpětím dochází k vyššímu namáhání ložisek. Důsledkem je přehřívání a snížení životnosti.

Vzhledem k účinkům tření a vysokým rychlostem je nutné výkonnější vřetena chladit. Nejúčinnějším způsobem je vodní chlazení. Nutnost chlazení vřetene v rámci této práce není nutná pro nízké obráběcí rychlosti.

Teoretickou hodnotu maximální rychlosti[19] lze vypočítat ze vzorce 3.1 kde SLF je konstanta SLF dle typu ložiska, ID je průměr vnitřního kroužku a OD je vnější průměr.

$$RPM_{MAX} = \frac{SLF}{(ID + OD)/2} = \frac{450000}{(8 + 22)/2} = 30000RPM \tag{3.1}$$

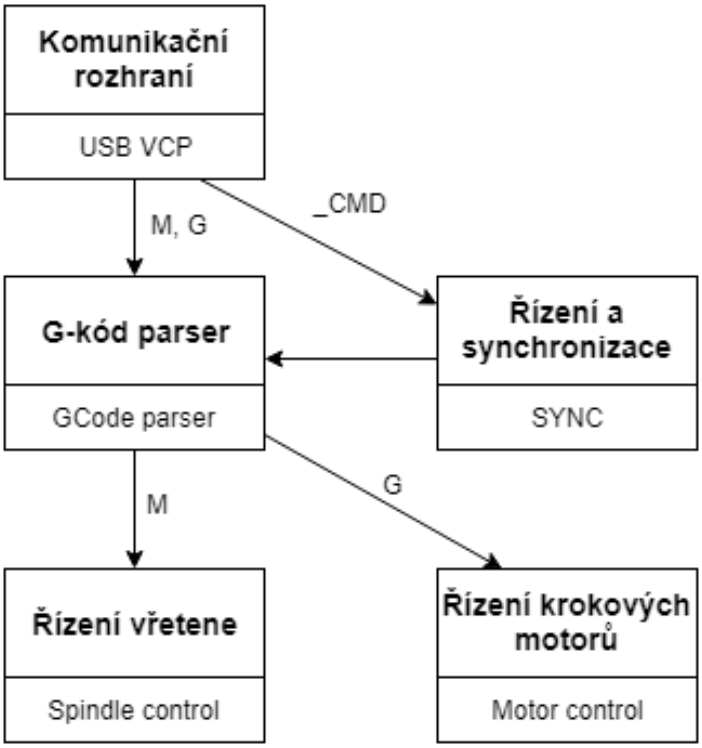
Z důvodu absence chlazení je maximální rychlost programově omezena na 10000 RPM.

3.2 Řídicí část

Pro řízení byly navrženy 3 možné řešení. Rozdíl je hlavně v rychlosti a rozšiřitelnosti celého zařízení. Hlavními částmi jsou synchronní řízení motorů, interpolátor a uživatelské rozhraní pro interakce s uživatelem.

3.2.1 Řešení s STM32

První návrh uvažuje řízení pomocí procesoru architektury ARM. Konkrétně se jedná o STM32F407 běžícím na frekvenci 168 MHz. V procesoru je nasazen realtimový operační systém FreeRTOS. Systém je připojen k PC pomocí virtuálního sériového portu. Z časovačů jsou vyvedeny signály pro generování kroků. Toto řešení je velmi omezené v rozšiřitelnosti.
Na následujícím obrázku 3.3 je blokový návrh celého systému. Každý blok představuje jednu úlohu (jeden task dle názvosloví RTOS). Pro komunikaci mezi úlohami jsou použité fronty, po kterých se předávají jednotlivé zprávy. Struktura zprávy je: data, příkaz a status. Přístup na komunikační rozhraní je blokován semaforem, pro možnost přístupu z ladících částí.



Obr. 3.3: Blokový návrh systému pro FreeRTOS na STM32

Na vstupu je komunikační rozhraní USB VCP. Jedná se o virtuální sériový port na standardu USB. Tento blok přijímá data z nadřazeného systému a informuje ho o stavu zařízení. Probíhá základní převzetí dat, jednoduché zpracování do struktury a verifikace podle kontrolního součtu. Dále je nutné rozhodnout, jestli se jedná o řídící zprávu začínající řetězcem "_CMD"nebo o část programu začínající písmeny 'M' nebo 'G'. Pro komunikaci s dalšími úlohami je vytvořeno univerzální rozhraní. To je důležité pro možnost nahrazení tohoto bloku libovolným jiným vstupem, například IP pro ovládání zařízení přes Ethernet nebo USB Mass Storage pro načítání programů z externího úložiště. Dle typu zprávy je dále přeposílána do řídící úlohy nebo parsovací úlohy. Parser zpracuje jednotlivé přijaté znaky a vypočítá vzdálenosti o které se mají jednotlivé osy posunout. Výpočty probíhají dle globálního konfiguračního souboru, který obsahuje konstanty jako stoupání šroubů, počet kroků na jednu otáčku motoru, zvolené mikrokrokování a jiné. Po výpočtu jsou data zabalena a přeposlána do úlohy řízení motoru nebo řízení vřetene.

Pro každý motor je vytvořen samostatný čítač, jehož periodou se ovlivňuje rychlost motoru. Počet pulzů udává ujetou vzdálenost v rámci jedné osy. Znaménko u hodnoty kroků určuje nastavení výstupu DIR, který ovládá směr otáčení motoru. Tato část je kritická pro přesnou synchronizaci. Čítače lze synchronizovat nastavením do režimů master-slave, ale tento přístup není dostatečně flexibilní a jeho rozšiřitelnost je omezená, a proto nebude realizován.

Řízení otáček vřetene probíhá pomocí čítače v módu PWM, který generuje servo signál pro modelářský regulátor. Otáčky jsou upravovány podle zpětné vazby z otáčkoměru. Vstup z otáčkového senzoru je připojen na čítač, který běží v módu čítače a měří periodu signálu.

3.2.2 Řešení ZYNQ + Linux

Poslední návrh využívá otevřené distribuce Linuxu upravené přímo pro ZYNQ. Velkou výhodou tohoto řešení je možnost sloučit grafické rozhraní a interpolační část. Slabinou tohoto řešení je potřeba řízení v reálném čase. Do FPGA lze zařadit zásobník (buffer), který bude cyklicky načítat data z linuxové části. Ale toto řešení nemusí stačit pro velké toky dat. Další možností je využití RT řízení přímo na procesoru. Systém umožňuje rozdělit výkon dvoujádrového procesoru pro kritické a nekritické aplikace Existují 2 přístupy, kterými lze dosáhnout RT řízení[20]:

- Nativní Linux - využívá vestavěné funkce linuxu, případně modifikování jádra (optimalizované pro nízkou latenci), RT úlohy se vytváří pomocí Linux API, zahrnuje 3 další možné přístupy: SMP (symetrický multiprocesing), Preemptivní RT a Enea LWRT (průmyslové řešení)
- Asymetrický multiprocesing - využívá každé jádro procesoru samostatně:

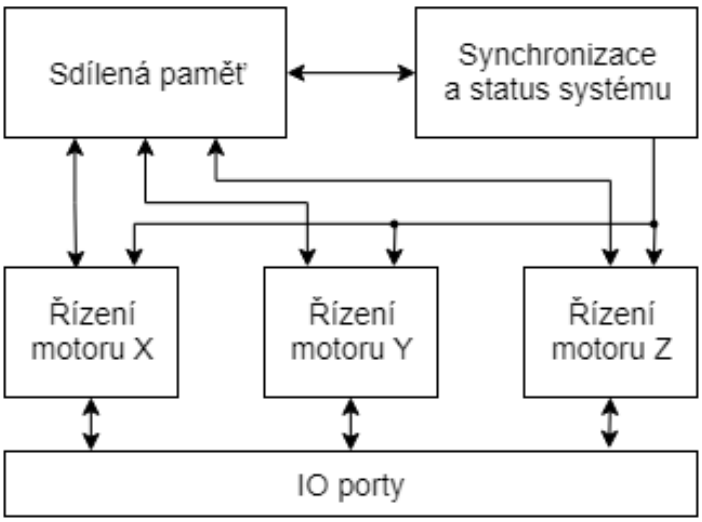
Linux + Bare-metal nebo Linux + FreeRtos; Bare-metal znamená, že program běží přímo na jádře procesoru bez jakéhokoliv OS

Toto řešení je nejideálnější z toho důvodu, že vše běží na jednom čipu. Stroj proto nemusí být doplněn počítačem.

3.2.3 Řešení ZYNQ + C/C++ program

Toto řešení představuje periferii k PC připojenou přes USB. RT systém je složený z jednotlivých úloh viz. předchozí návrh pro STM32 3.3 nebo může být obsluhován hlavní smyčnou programu. Rozdíl je v procesoru, na kterém běží a ve způsobu řešení nejnižší vrstvy. Program běží na jednom jádře procesoru ARM v rámci SoC ZYNQ. Kód je dobře přenositelný z předchozího návrhu díky shodným knihovnám. HW vrstva tvořená čítači v předchozím návrhu je nahrazena vlastní AXI periferií vytvořenou pro hradlové pole v ZYNQu. Na tuto verzi bylo přistoupeno z důvodu nedostatečnosti návrhu s STM32, a to zejména kvůli synchronizaci mezi motory, rozšiřitelnosti systému a rychlosti systému.

Na obrázku 3.4 je návrh blokového schématu AXI periferie.



Obr. 3.4: Blokový návrh systému pro FPGA na platformě ZYNQ

3.2.4 Řízení otáček vřetene

Driver pro motor vřetene je řízen servo signálem o frekvenci 50 Hz s délkou pulzu 1 ms pro nulové otáčky a 2 ms pro maximální otáčky. Otáčky jsou snímány pomocí optického snímače na hřídeli sklíčidla. Pro řízení otáček je nutné navrhnout regulátor, který bude vyhodnocovat odchylku aktuální rychlosti od požadované rychlosti a

Tab. 3.1: Hodnoty parametrů dle Ziegler - Nicholsonovy metody

regulátor	Kp	Ti	Td
P	$P_{KRIT}/2$	-	-
PI	$P_{KRIT}/2.2$	$T_{KRIT}/1.2$	-
PID	$P_{KRIT}/1.7$	$T_{KRIT}/2$	$T_{KRIT}/8$

upravovat střídu PWM výstupu. Ideálním řešením je diskrétní PID regulátor, který je tvořen třemi složkami: proporcionální, integrační a derivační. Matematický popis diskrétního regulátoru je ve vzorci 3.2.

$$y(t) = K_p[e(t) + \frac{1}{T_i}\sum_{t=0}^{t} e(t) + T_d(e(t) - e(t-1))] \tag{3.2}$$

kde:

- K_p zesílení [-]
- T_d časová derivační konstanta [s]; definovaná jako $\frac{K_p}{I}$
- T_i časová integrační konstanta [s]; definovaná jako $\frac{K_p}{D}$

Výsledkem je reakční zásah, který je nutné aplikovat na akční člen. Proporcionální složka K_p zahrnuje přímý výsledek násobení P konstanty a regulační odchylky. Při zvolení vysoké hodnoty může dojít k nestabilitě systému a jeho rozkmitání. Integrační složka je integrál regulační odchylky násobené I konstantou. Tato složka zajišťuje dosažení nulové regulační odchylky. Derivační složka stabilizuje celý proces, snižuje překmit a dobu přechodu.

Důležitou částí je anti-windup filtr. V případě, že motor není schopný dosáhnout zadaných otáček ani maximálním možným akčním zásahem na vstupu regulátoru, docházelo by ke zvyšování hodnoty v integrátoru. To by se projevilo při další změně otáček, která by trvala dlouho.

Pro ladění se používá Ziegler-Nicholsonova metoda. Prvním krokem této metody je zjištění kritického zesílení P. Postupným zvyšováním konstanty K_c je nalezeno zesílení, pro které systém netlumeně kmitá. Toto zesílení je označeno jako kritické P_{KRIT}. Perioda kmitů pro toto zesílení je označena jako kritická T_{KRIT}. Hodnoty parametrů jsou následně nastaveny podle tabulky 3.1 [21].

3.2.5 Parser G-kódu

Vstupními daty jsou soubory, které obsahují G-kód v prostém textu. Text je nutné zpracovat do struktur, které bude možné dále v systému zpracovat. Pro zpracování G-kódu napsán vlastní parser. Jednotlivé příkazy ze zdrojového souboru jsou rozpoznávány v řídícím programu s využitím regulárních výrazů.

3.2.6 Post procesor

Pro zajištění podpory CAM softwaru Cut2D a dalších byl navržen konfigurační soubor pro postprocesor. Konfigurační soubor je v repozitáři projektu. Struktura byla navržena dle příručky softwaru[22]. V souboru byla nastaveny následující parametry:

- přípona výstupních souborů - "*.gcode"
- základní jednotky - milimetry
- ukončování řádků - [13][10]
- číslování řádků a rychlost vřetene - celočíselně
- řezná rychlost - 1 desetinné místo
- souřadnice - 3 desetinné místa

Pro samotný program byly definována hlavička, podporované příkazy a patička. V hlavičce se nachází posloupnost příkazů "G17 G21 G91"zajišťující nastavení stroje pro obrábění v rovině XY, jednotku na milimetry a přírůstkový režim. V podporovaných kódech je definována rychlá interpolace $RAPID\_MOVE$ jako G0, první řezný pohyb $FIRST\_FEED\_MOVE$ jako G1 se zadáním parametru rychlosti a ostatní řezné pohyby $FEED\_MOVE$ jako G1 jenom se souřadnicemi.

3.2.7 Komunikační protokol

Pro komunikaci přes VCP byl navržen jednoduchý komunikační protokol. Protokol je nezávislý na nižších vrstvách. Lze jej proto použít i při komunikaci přes soket. Struktura rámce a jednoduchý příklad jsou následující:

```
{STX}příkaz;argument1;argument2;argument3...{ETX}{CR}{LF}
"\u0002G_CODE;X10;Y5.5;Z0\u0003\r\n"
```

Znak STX (0x02 v ASCII) je označení začátku rámce. Následuje příkaz který vyjadřuje význam rámce. Příkaz může ale nemusí být doplněn argumenty. Ty jsou mezi sebou odděleny oddělovačem jak lze vidět na příkladu. Jako oddělovač je zvolen středník. Argumenty můžou obsahovat písmena, čísla, tečku a čárku. Validní formy argumentu jsou:

- celočíselný - 25
- desetinný - 2.5 nebo 2,5
- pojmenovaný celočíselný - X10

- pojmenovaný desetinný - RAD15.5 nebo RAD15,5
- prostý text - INFO

Pro desetinné typy je nutné uvažovat tečku i čárku jako desetinný oddělovač. Protokol tedy není závislý na systému na kterém je použit. Rámec je ukončen znakem *ETX* (0x03 v ASCII). Příkaz může být doplněn znaky pro zalomení řádku - CR a LF. Tato možnost je určená primárně pro testování, kdy je komunikace s využitím konzolové aplikace uživatelsky přívětivější.

Protokol je určen pro použití požadavek-odpověď. Nadřazený systém začíná komunikaci a vysílá požadavek. Podřízený systém musí odpovědět do určité doby (nadřazený systém čeká 500 milisekund na odpověď). V tabulce 3.2 je výčet implementovaných příkazů:

Tab. 3.2: Implementované příkazy

Požadavek	**Odpověď**	**Popis**
ECHO	ECHO	Ověření komunikace
START	START	Zapnutí zpracovávání příkazů v zásobníku
STEP	STEP	Krokování příkazů v zásobníku
STOP	STOP	Zastavení zpracování příkazů v zásobníku
RESET	RESET	Potvrzení chyby
G_CODE	G_CODE	Odeslání G-kódu
GET_POS	GET_POS;0;0;0	Získání aktuání pozice
ZERO;X	ZERO	Vynulování osy, hodnoty: X,Y,Z,ALL
SET_SPD_MULT	SET_SPD_MULT	Nastavení násobitele rychlosti 0-1
RUN_SPINDLE	RUN_SPINDLE	Spuštění vřetene [otáčky/minutu]
STOP_SPINDLE	STOP_SPINDLE	Zastavení vřetene
DISP_TEST	DISP_TEST	Test dávkovače

3.3 Hardwarová část

3.3.1 Vývojový kit

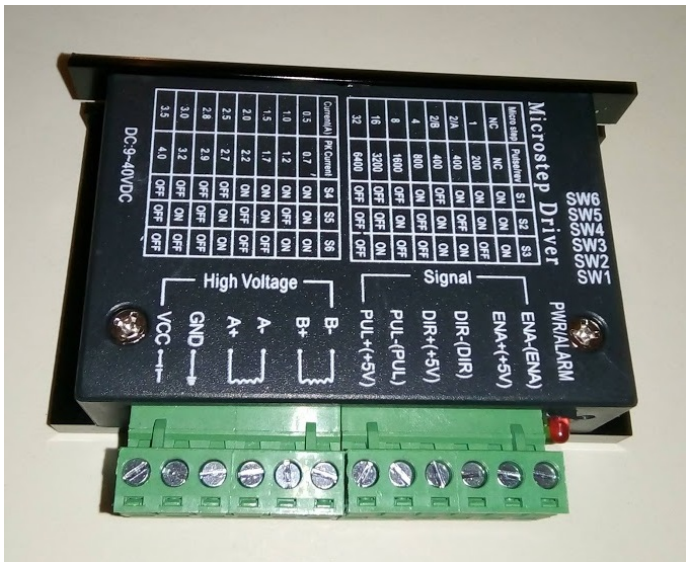
Hlavní řídící částí je vývojový kit ZYNQ Zybo Trainer Board[24], který je osazen čipem XC7Z010-1CLG400C. Napájecí zdroj pro vývojový kit má výstupní napětí 5V a proud 3A. Pro základní testování bez periferií lze kit napájet z USB 2.0 (dle specifikace poskytuje proud maximálně 500mA). Vyšší proud zdroje je doporučen výrobcem. Desku je nutné doplnit rozšiřující deskou pro připojení driverů.

3.3.2 Krokové motory

Pro výbornou dostupnost a špičkové parametry byl pro řízení zvolen krokový motor SX23-1414. Motory je možné zapojit jako unipolární, bipolární sériové a bipolární paralelní. Jmenovitý moment motoru je 1,4 Nm. Velikost kroku je 1,8 ° tzn. že motor má 200 kroků na jednu otáčku. Jmenovitý proud motoru je 1,4 A pro sériové zapojení a 2,8 A pro paralelní.

3.3.3 Drivery pro krokové motory

Pro řízení byly zvoleny drivery TB6600 včetně chladiče a galvanického oddělení na vstupu viz. 3.5. Maximální proud driverů je 4A. Maximální napájecí napětí je 40V. Pro zvolené motory jsou naddimenzované, aby bylo možné je nahradit silnějšími. Drivery jsou napájené ze zdroje 24V 10A.



Obr. 3.5: Driver TB6600

Nastavení proudu motorem a mikrokrokování lze nastavit pomocí DIP přepínačů. Pro testovací účely bylo zvoleno mikrokrokování 1/4 a proud 400mA. Tento proud je dostatečný pro pohyby os a nedochází k nadměrnému zahřívání motorů. Hodnota mikrokrokování byla zvolena z důvodu plynulosti běhu motoru.

3.3.4 Motor pro pohon vřetene

Pro vřeteno byl zvolen bezkartáčový motor BC-3530 o výkonu 460W. Napěťová konstanta motoru je 1700 ot./V. Pro napájení bylo zvoleno napětí 12 V. To odpovídá maximálním otáčkám 20400 otáček/min. Maximální otáčky budou muset být programově omezeny dle maximálních otáček samotného vřetene.

3.3.5 Driver pro motor poháněj́ící vřeteno

Pro řízení motoru vřetene byl vybrán 40A driver viz obrázek 3.6. Tento driver je určený pro RC modely napájené z 2-3 ks lithium-polymerových akumulátorů. Tomu odpovídá napětí 7,4 nebo 11,1 V.

Obr. 3.6: Driver SKYWALKER 40A

Vstupním signálem je servo signál o frekvenci 50 Hz a délce pulzu 1 - 2 ms pro 0 - 100 % otáčky. Driver zajišťuje řízení motoru pomocí proudové regulační smyčky. Externí řídící systém může ovládat pouze rychlost.

Tento driver byl zvolen z důvodu dostupnosti a jednoduchosti použití. Do finálního stroje nelze toto řešení použít z důvodu absence reverzace otáček a nemožnosti regulovat proud motorem dle specifických požadavků.

4 Realizace řídicí části - ARM

Tato část je psána v jazyce C++ dle standardu C++14. Cílový procesor pro tuto práci je ARM Cortex A9. Jednotlivé knihovny jsou přizpůsobeny pro snadnou přenositelnost na jiné procesory. Základem je program složený z jednotlivých knihoven a oddělující vrstvou pro části závislé na procesoru. Jedná se o části jako je třeba přístup k paměti nebo sériové rozhraní. V hlavní smyčce programu je obsluhován primárně příjem dat z nadřazeného systému a cyklické zpracovávání jednotlivých příkazů.

Tato část je vyvýjena ve dvou projektech. V rámci projektu $cnc-app$ jsou vyvýjeny a testovány jednotlivé knihovny celého systému, které nejsou závislé na cílové platformě. V rámci stromové struktury projektu je vytvořena složka pro zdrojové kódy *src*, pro externí knihovny *lib* a pro testovací procedury *test*. Překlad probíhá pomocí nástroje CMake. Ten je nakonfigurován pomocí konfiguračních souborů $CMakeLists.txt$. Díky tomuto nástroji lze knihovny přeložit na operačním systému Linux i Windows. Projekt lze doplnit libovolnou externí knihovnou, kterou lze přeložit pomocí CMake. Do projektu je přidána knihovna pro testování - CppUTest. Samotné použití je popsáno v kapitole Testování kódu.

Druhým projektem je projekt $cnc-simple$, který je upravován a překládán s pomocí nástroje Xilinx SDK. Tento projekt obsahuje jednoduché komunikační rozhraní a přístup do paměti. K této části jsou připojeny knihovny z projektu zmíněného výše. Dále je tento nástroj použit pro ladění samotného programu.

Každá knihovna je definována rozhraním v které jsou abstraktně vyjádřeny hlavní metody. Například pro řízení nástroje je potřeba časovač. Ten je v produkčním kódu implementován pomocí knihoven od Xilinx. Pro testy je implementována redukovaná verze $TimerControlMock.hpp$ ve kterém jsou implementovány metody dle rozhraní bez vnitřní logiky. Dále jsou argumenty jednotlivých metod ukládány do veřejných proměnných. Je to z důvodu testování, kdy je nutné ověřit jestli do nižší vrstvy byly předány korektní data.

V následujících kapitolách jsou popsány jednotlivé knihovny celého systému. V každé části je také popsáno jak byla otestována jejich funkčnost. V poslední kapitole je popsán proces automatického testování kódu.

4.1 Přístup k paměti

Přístup k paměti může být odlišný pro jednotlivé systémy. Z toho důvodu bylo pro tento účel vytvořeno rozhraní *IMemmoryAccess*. Jednotlivé nadřazené třídy tedy neví k čemu přistupují a jsou proto nezávislé na cílové platformě. AXI periferie jsou v hradlovém poli paměťově mapované - lze k přístupu k jejich registrům využít právě toto rozhraní.

```
class IMemoryAccess
{
public:
    virtual uint32_t ReadRegister(uint32_t offset) = 0;
    virtual void WriteRegister(uint32_t offset, uint32_t data) = 0;
    virtual ~IMemoryAccess() = default;
};
```

Rozhraní definuje metody *ReadRegister* pro čtení dat z registru, *WriteRegister* pro zápis dat do registru a výchozí destruktor. Výchozí destruktor lze předefinovat v implementaci například nastavením výchozích hodnot do registrů. Parametr $offset$ určuje posun v bytech. Jednotlivé třídy, které implementují toto rozhraní, jsou silně závislé na cílovém procesoru. V projektu jsou vytvořeny 2 implementace. První je určena pro testovací účely a druhá je určena pro reálný procesor.

První implementace je využívána pro testovací účely. Lze ji ale využít i pro některé systémy kde jsou části sdílené paměti definovány přímo adresou.

```
uint32_t MemoryAccess::ReadRegister(uint32_t offset)
{
    CheckOffset(offset);
    return *(_reference + offset);
}
void MemoryAccess::WriteRegister(uint32_t offset, uint32_t data)
{
    CheckOffset(offset);
    *(_reference + offset) = data;
}
```

Do konstruktoru je předána výstupní reference, která označuje začátek periferie, a její délka. Délka je nutná pro zamezení přístupu mimo vyhrazenou paměť periferie. Při zápisu i při čtení je zavolána metoda CheckOffset, které kontroluje validitu argumentu offset. Tento offset musí být nižší než zadaná délka a zároveň musí být dělitelný číslem 4 - kontrola zarovnaného přístupu. Je to z důvodu nutnosti přístupu k paměti po celé šířce - 4 bytech. Při nesprávném použití dojde v vyvolání globální výjimky *DataAbort* přímo v procesoru, která značí nesprávný přístup do paměti.

V rámci jednotlivých testů je proto možné plně otestovat funkčnost této vrstvy. A to zejména základní funkci zápisu a čtení v případě správně i nesprávně zadaného

argumentu offsetu. V případě pokusu o nezarovnaný přístup nebo přístup mimo paměť je vyvolána výjimka $logic_error$ protože se jedná o nesprávné použití dané knihovny. Předchází se tak vyvolání výjimky *DataAbort* u které by se složitěji určovalo kde k chybě došlo (výjimka se odchytává v nejvyšší vrstvě programu).

4.1.1 Implementace pro reálný procesor

Implementace je upravená pro cílový procesor využitím metod Xil_In32 a Xil_Out32 z knihovny *xil_io.h*. Tato knihovna obsahuje vstupně výstupní funkce které přistupují přímo k registrům.

```
uint32_t XMemoryAccess::ReadRegister(uint32_t offset)
{
    CheckOffset(offset);
    return Xil_In32((_baseAddress) + (offset));
}
void XMemoryAccess::WriteRegister(uint32_t offset, uint32_t data)
{
    CheckOffset(offset);
    Xil_Out32((_baseAddress) + (offset), (u32)(data));
}
```

Do konstruktoru je předána base adresa a délka periferie. Kontrola vstupního ofsetu je kontrolována stejně jako u implementace pro testy.

4.2 Řízení rokového motoru

Tento blok představuje řadič pro AXI periferii řízení motoru. Periferie je paměťově mapovaná. Z toho důvodu je nutné použít Přístup do paměti. V hlavičkovém souboru jsou definovány jednotlivé offsety registrů odpovídající HW implementaci v FPGA.

```
enum RegisterOffsets
{
    RequiredPosition = 0,
    ActualPosition = 4,
    Speed = 8,
    Acceleration = 12,
    ModeOfControl = 16,
    Commands = 20,
    Status = 24
};
```

Pro registry *Commands* a *Status* jsou definovány offsety pro jednotlivé bity. Krokový motor může pracovat ve třech módech: poziční řízení, rychlostní řízení a kalibrace. Výčet módů a metody pro práci s nimi lze vidět v následujícím příkladu:

```
enum StepperMode
{
    POSITION_CONTROL = 0,
    SPEED_CONTROL = 1,
    CALIB = 2
};
void SetMode(StepperMode mode);
StepperMode GetMode();
```

Dále jsou implementovány metody pro práci s pozicí $SetPosition(uint32\_tposition)$ a $GetActualPosition()$. V této úrovni je jednotkou pozice krok. Pro nastavení rychlosti je definována funkce $SetSpeedPeriod(uint32_tperiod)$ kde parametr period představuje periodu výstupního signálu. Synchronizační pulz pro spuštění operace zajišťuje metoda $GenerateSyncPulse()$. Řízení synchronizačního pulzu je rozdílné pro poziční a rychlostní řízení. Při pozičním rychlostí je generován pouze pulz pro spuštění oparace. Při rychlostním řízení se motor pohybuje dokud je synchronizační pulz aktivní. Vnitřní čítač pozice lze vynulovat pomocí metody $ResetCounter()$. Dále lze ovládat pin Enable přímo na driveru - je umožněno zapnutí a vypnutí driveru. Pro možnost práce s registry po bitech byly přidány metody $SetBit()$ a $ResetBit()$.

4.3 Řízení nástroje

Je to knihovna určená pro řízení nástroje. V aktuální verzi je knihovna připravena pro řízení vřetene a dávkovače. Pro tento účel jsou potřeba 2 čítače. Vřeteno využívá 2 čítače: generování PWM a měření otáček. Pro dávkovač je využit pouze jeden čítač pro vygenerování pulzu.

Výčtový typ enum $ToolControlMode$ definuje dva možné režimy ve kterém může tato třída pracovat: Spindle (režim pro vřeteno) a Dispenser (režim pro dávkovač). Režim lze přepínat pomocí metody $SetMode(ToolControlModemode)$, ale pouze když neběží vřeteno nebo neprobíhá dávkování. V opačném případě by došlo k vyvolání výjimky $logic\_error$.

Vřeteno lze spustit pomocí metody $StartSpindle(uint16\_trpm)$ do které se předává parametr rpm - otáčky za minutu.

Dávkovač se ovládá metodou $Dispense(uint16\_ttimeMiliseconds)$ do které se jako parametr předává čas v milisekundách.

Řízení vřetene je doplněno regulátorem PID popsaným v teoretické části. Regulátor je definován rozhraním *IPid*.

```
class IPid
{
public:
    virtual double Calculate(double setValue, double actualValue) = 0;
    virtual ~IPid() = default;
};
```

V implementaci tohoto rozhraní jsou v konstruktoru předány základní parametry P, I a D. Ty jsou uloženy do veřejných proměnných z důvodu možnosti změny parametrů za chodu. Dále je předána perioda obnovovací smyčky a meze pro výstupní signál. Perioda nesmí být nulová. To je nutné hlavně pro ladění. Bezpečnost změn jednotlivých konstant musí být zajištěna ve vyšších vrstvách. Při nesprávném použití nelze zamezit poškození aktuátoru, který je na výstupu z regulátoru.

4.4 Řízení pohybu

Představuje skupinu os, které jsou řízeny dle zadaných souřadnic v kartézském systému. Je definován rozhraním *IMovementControl*:

```
class IMovementControl
{
public:
    virtual void MoveAbsolute(
        uint32_t x,
        uint32_t y,
        uint32_t z,
        uint32_t speed) = 0;
    virtual void MoveRelative(
        int32_t dx,
        int32_t dy,
        int32_t dz,
        uint32_t speed) = 0;
    virtual void SetHomePostion(AxisSelector axis) = 0;
    virtual ~IMovementControl() = default;
};
```

Pomocí metody *MoveAbsolute*(...) je možné pozicovat osu na absolutní pozici. *MoveRelative*(...) umožňuje pohyb relativně k aktuální pozici. Pozice vychází z pozice vyčtené z periferií v hradlovém poli. Metoda *SetHomePosition*(*AxisSelectoraxis*) umožnuje vynulovat pozice jednotlivých os. Parametr *AxisSelector* určuje osu která má být vynulována.

Konstruktor implementace tohoto rozhranní je v následující části hlavičkového spouboru *MovementControl.hpp*.

```
explicit MovementControl(
    std::unique_ptr<IStepperMotor> stepperMotorX,
    std::unique_ptr<IStepperMotor> stepperMotorY,
    std::unique_ptr<IStepperMotor> stepperMotorZ);
```

4.5 Zpracování G-kódu

Třída pro zpracování jednotlivých G-kódů a M-kódů. Vstupem je příkaz z komunikační části. Dle typu je řízen pohyb os, vřeteno nebo dávkovač. Při vytváření instance jsou předány třídy: řízení nástroje a řízení pohybu. Pokud je příchozí příkaz G-kód následuje přepočet z milimetrů na mikrokroky dle konstant udevedývh v konfiguračním souboru. Výsledné hodnoty jsou předány do řízení pohybu. Volá se funkce *MoveAbsolute* nebo *MoveRelative* dle hodoty proměnné *absoluteActive*, která určuje jestli mají být zadané příkazy zpracovány v absolutním nebo přírůstkovém režimu. Hodnotu této proměnné lze změnit příkazem *G*90 (absolutní režim) a *G*91 (přírůstkový režim). Příkazy pro nástroj (M3, M5 a M77) jsou předány do řízení nástroje. Příkaz M3 je pro zapnutí a M5 pro vypnutí vřetene. Příkaz M77 je specifický pro tento druh stroje - jedná se o řízení dávkovače.

4.6 Testování kódu

Pro ověření funkčnosti jednotlivých částí byl použit testovací framework CppUTest [25]. Další výhodou testování je možnost vývoje bez HW části. Nejnižší vrstvy které jsou závislé na procesoru se nahradí implementací která simuluje chování HW a je možné vůči ní pouštět testy. Velkou výhodou tohoto procesu je úspora času při vývoji.Minimalizuje se četnost nahrávání programu do procesoru a také nutnost použití ladění. V případě rozsáhlého projektu je možné chybou v jedné části systému možné ovlivnit chování několika ostatních částí - situace kdy práce s ladícím nástrojem trvá déle než napsání samotných testů. Celý proces je následně možné zautomatizovat na testovacím serveru. Při vývoji této práce byl využit server GitLab, který nabízí sdílené prostředky pro CI - Continuous Integration. Jedná se o službu která spouští jednotlivé úlohy definované uživatelem na sdílených serverech. Při nahrání kódu na server se automaticky spustí zadaná úloha - Pipeline. Pipeline je tvořena jednotlivými skripty, které lze uspořádat do jednotlivých fází. Úloha je nakonfigurována pro sestavování programu a spouštění testů. Tento projekt má tedy 2 fáze: sestavení(Build) a testování(Test). Konfigurace jednotlivých částí automatizovaného procesu je v souboru $.gitlab - ci.yml$.

5 Realizace řídicí části - FPGA

5.1 AXI periferie pro řízení osy

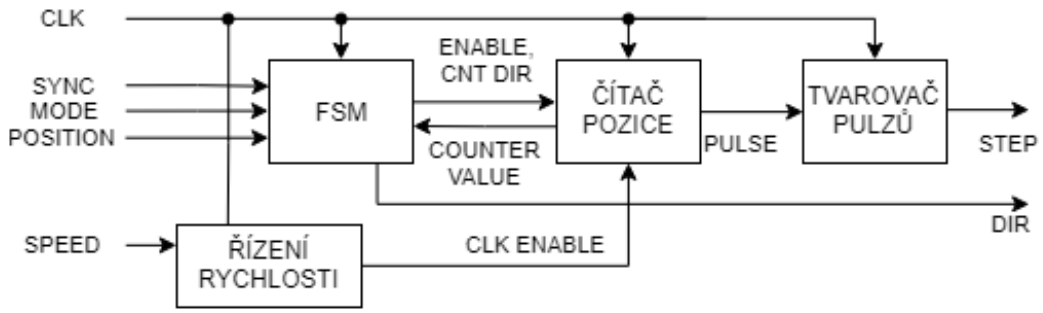
AXI periferie je paměťově mapovaná. Rozhraním mezi systémy je sdílená paměť, která obsahuje jednotlivé řídící struktury. Veškeré funkce jsou ovládány a kontrolovány pomocí registrů: datových, příkazových a statusových. Do datových se zapíšou parametry vyžadované operace, příkazovým dojde ke spuštění a z hodnoty status registrů je kontrolován výsledek operace. Výčet registrů jedné instance řízení motoru je v tabulce 5.1.

Tab. 5.1: Mapování registrů pro řízení jednoho motoru

Offset adresy	Registr	Typ registru
0x0000	Požadovaná pozice	Datový
0x0004	Aktuální pozice	Statusový
0x0008	Rychlost	Datový
0x000C	Zrychlení	Datový
0x0010	Mód řízení	Datový
0x0014	Příkazy	Příkazový
0x0018	Status	Statusový
0x001C	Rezerva	-

Požadovaná pozice je vstupní registr pro nastavení cílové pozice. Zapisuje se hodnota v krocích/mikrokrocích - hodnota odpovídá počtu pulzů, které jsou vygenerovány na výstupu. Aktuální pozice - je status registr, který udává aktuální pozici v krocích. Po dokončení pozicování se hodnoty prvních dvou registrů rovnají. Rychlost je zadávána jako počet nanosekund mezi jednotlivými pulzy - rozlišení systému je nastaveno na 10 ns. Registr zrychlení udává počet nanosekund, které jsou postupně odčítány od základní periody 100 ms až po dosažení zvolené rychlosti. Správný výpočet je zajištěn ve vyšším systému. Mód řízení je rychlostní, poziční nebo kalibrační. Rychlostní je určen pro tzv. jog mód, kdy se osa pohybuje, dokud je stisknut ovládací prvek. Poziční systém je využívám při zpracovávání jednotlivých příkazů. A kalibrační mód zajišťuje nastavení nulových pozic jednotlivých os - automaticky dojetím na koncové spínače, nebo manuálně při požadavku na posun nuly stroje. Registr příkazy zajišťuje spouštění bloku. Obsahuje startovací bit, kterým je možné spustit běh motoru, a další 2 bity, které ovládají kalibraci (spuštění a volba režimu automatická/manuální).

Na následujícím obrázku 5.1 je blokové schéma části řízení motoru. Do projektu jsou vloženy celkově 3 instance - pro motory X, Y a Z. Na vstupu je stavový automat FSM, který řídí běh čítače pozice. Jednotlivé stavy budou popsány níže. Blok čítač pozice uchovává aktuální pozici. Je tvořen čítačem spouštěným signálem ENABLE. Frekvence čítání se získává z bloku řízení rychlosti a směr je udáván signálem CNT DIR. Signál udávající aktuální pozici je veden zpět do bloku FSM. Blok tvarovač pulzů upravuje výstupní signál. Nastavení tohoto bloku je závislé na zvoleném driveru (nastavuje se dle časového diagramu uvedeném výrobcem).

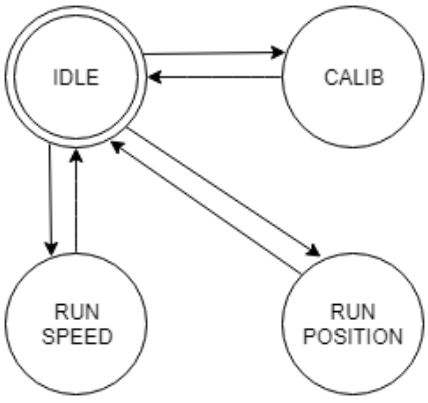


Obr. 5.1: Blokové schéma řízení motoru

Stavový automat pro řízení motoru obsahuje 4 možné stavy:

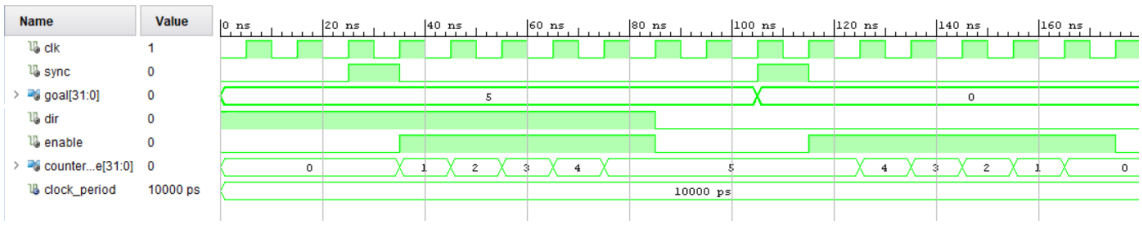
1. IDLE - nečinnost, výchozí stav
2. CALIB - kalibrace nastavující výchozí hodnotu do čítače
3. RUN SPEED - rychlostní řízení
4. RUN POSITION - poziční řízení

Mezi stavy jsou možné přechody vyznačené ve stavovém diagramu 5.2.



Obr. 5.2: Stavový diagram řízení motoru

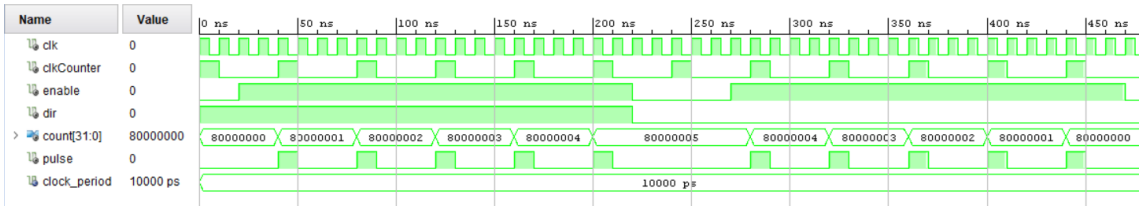
Byl zvolen Mealyho typ - následující stav závisí na aktuálním stavu a vstupech. Na obrázku 5.3 je simulace stavového automatu pro poziční mód.



Obr. 5.3: Simulace stavového automatu

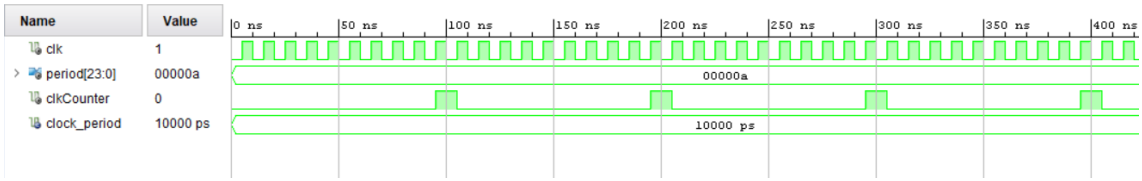
Na začátku je nastavena požadovaná pozice-*goal*, které má být dosaženo. Na základě rozdílu požadované a aktuální pozice je vygenerován signál *dir*, který určuje směr čítání a zároveň směr otáčení motoru. Informace o aktuální pozici je získána z čítače o úroveň níže - signál *counterValue*. Celý proces začíná vygenerováním synchronizačního pulzu. Následuje nastavení logické jedničky do signálu *enable* tj. povolení čítání. Následuje simulace chování běžícího čítače. Hodnota čítače je zvyšována až do hodnoty požadované pozice. Následuje opačný proces simulující pohyb motoru na druhou stranu. V této části je ověřena správnost chování signálu *dir*.

Na stavový automat je napojen čítač pozice, který lze ovládat vstupy *enable*, *dir* a *clkCounter*. Signály *enable* a *clkCounter* povolují čítání a signál *dir* určuje jeho směr. Jeden povolující signál je generován z bloku řízení rychlosti a druhý ze stavového automatu. Simulace je na obrázku 5.4.



Obr. 5.4: Simulace čítače pozice

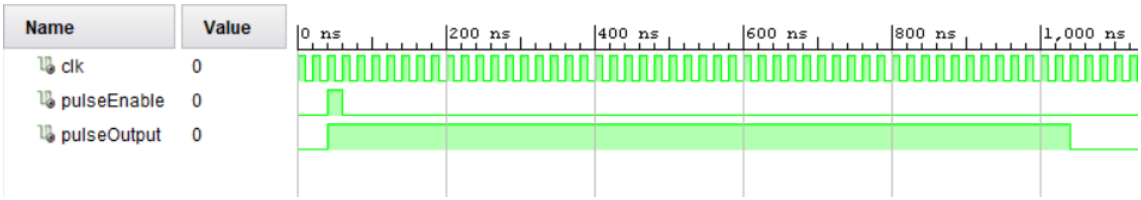
Blok řízení rychlosti je dělička kmitočtu s nastavitelnou periodou. Simulace tohoto bloku pro dělitel 10 je na obrázku 5.5.



Obr. 5.5: Simulace bloku řízení rychlosti

Posledním blokem je tvarovací obvod výstupních pulsů. Na obrázku 5.6 lze vidět simulaci tohoto bloku. Délka pulzu pro simulaci byla nastavena na 1 us. Z výsledku

simulace je zřejmé že funkce odpovídá návrhu.



Obr. 5.6: Simulace výstupního tvarovače pulzů

Při přivedení pulzu na vstup vygeneruje pulz pro budič motoru. Délka tohoto pulzu je ve finálním řešení zvolena dle použitého driveru. Minimální délka pulzu je vypočítaná dle vzorce 5.1.

$$t_{min} = \frac{1}{2}\frac{1}{f_{max}} = \frac{1}{40000} = 25us \tag{5.1}$$

5.2 AXI časovač pro řízení nástroje

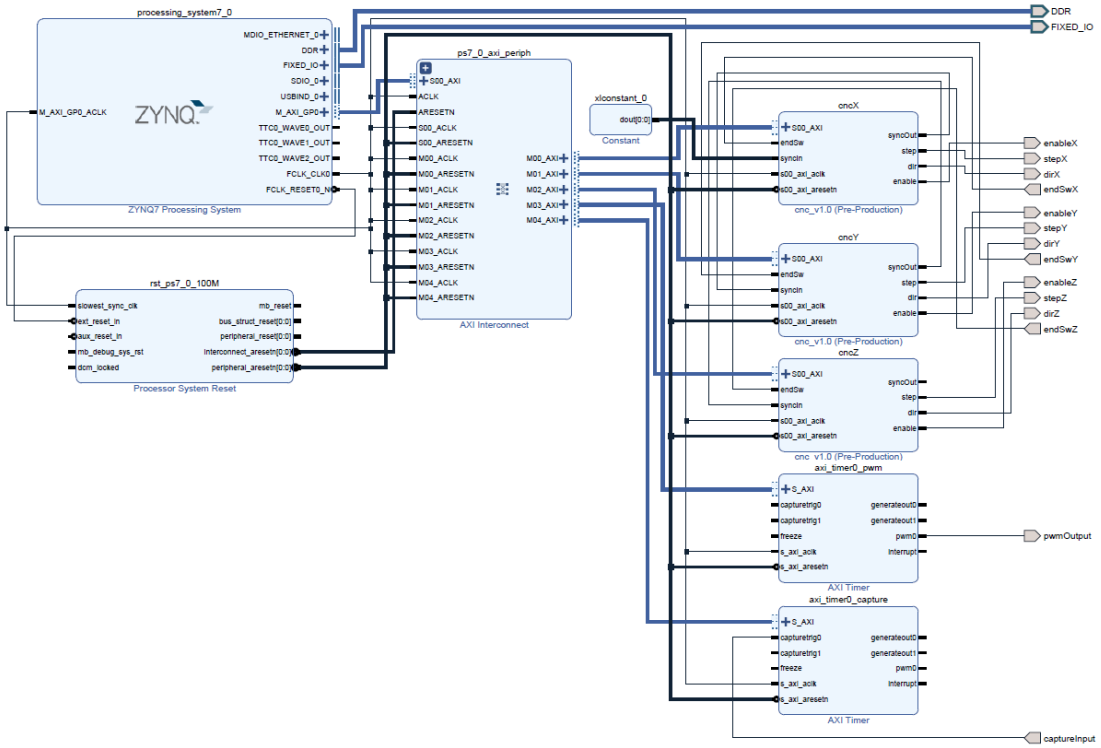
Na AXI sběrnici je připojen časovač pro potřeby řízení vřetene nebo dávkovače. »(následující přesunout do ARM části)Časovač může pracovat ve 2 módech: PWM a pulz. PWM mód je využíván pro řízení vřetene. Časovač generuje PWM signál o frekveni 50 - 500 Hz. Střídu je možné nastavit v rozmezí 5 - 10 %. Frekvence je zvolena dle možností použitého budiče bezkartáčového motoru. Změnou střídy lze měnit otáčky motoru.

Časovač je popsán v dokumentaci [26]. Periferie může pracovat v následujících módech: normální, zachycení události, pulsně šířková modulace (PWM) a kaskádový. Pro účely této práce je využit mód zachytávání vnější události a pulsně šířková modulace.

Pro nastavení módu PWM jsou využity obě poloviny periferie. První čítač určuje periodu výsledného signálu. Tu je možné vypočítat ze vstupní frekvence časovače, která je po vygenerování BSP(vysvetlit) uvedena v hlavičkovém souboru *xparameters.h* v konstantě $XPAR\_AXI\_TIMER0\_PWM\_CLOCK\_FREQ$. Pro aktuální aplikaci je frekvence 100 MHz. (přidat výpočet?) Druhý časovač určuje střídu signálu. Výsledná hodnota je poměrem period obou časovačů. (přesunout do ARM části)Směr čítání je dolů aby bylo možné měnit hodnotu v průběhu čítání. Při čítání nahoru by se musela hodnota měnit při dosažení hodnoty 0 aby nedošlo k chybně generovanému pulzu.

5.3 Blokové schéma

Tato kapitola popisuje zapojení jednotlivých AXI periferií na systém ZYNQ7. Blokové schéma je na obrázku 5.7.

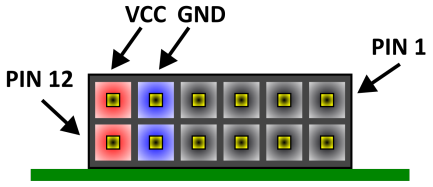


Obr. 5.7: Číslování konektoru PMOD

V projektu je vytvořeno blokový digram s hlavním systémem ZYNQ7 na který jsou s využitím vnitřního propojení napojeny jednotlivé instance řízení krokových motorů a časovače pro řízení vřetene/dávkovače.

5.4 Mapování

Jednotlivé vstupy a výstupy jsou namapovány na fyzické piny v souboru *Zybo − Master.xdc*. Jedná se o soubor obsahující veškeré piny vyvedené na vývojové desce ZYBO. Nepoužité piny jsou zakomentovány. Signály jsou přivedeny na konektory PMOD. Číslování konektoru odpovídá obrázku 5.8 dle referenčního manuálu [27].



Obr. 5.8: Číslování konektoru PMOD

Zapojení jednotlivých signálů je uvedeno v tabulce 5.2. Dle referenčního manuálu [27] jsou zvoleny vysokorychlostní porty PMOD-C a PMOD-D. Pro vstupy z koncových spínačů je nutné nastavit interní pull-down rezistor pokud jsou nevyužity. ochranné rezistry nejsou nutné z důvodu oddělení optočleny.

Tab. 5.2: Zapojení signálů

PMOD	Pin	Signál	Popis
JC	1	endSwX	Koncový spínač osy X
	2	dirX	Volba směru osy X
	3	stepX	Step pulzy pro osu X
	4	enableX	Aktivace osy X
	7	endSwY	Koncový spínač osy Y
	8	dirY	Volba směru osy Y
	9	stepY	Step pulzy pro osu Y
	10	enableY	Aktivace osy Y
JD	1	endSwZ	Koncový spínač osy Z
	2	dirZ	Volba směru osy Z
	3	stepZ	Step pulzy pro osu Z
	4	enableZ	Aktivace osy Z
	7	pwmOutput	Výstup PWM pro dávkovač/vřeteno
	7	captureInput	Vstup zpětné vazby

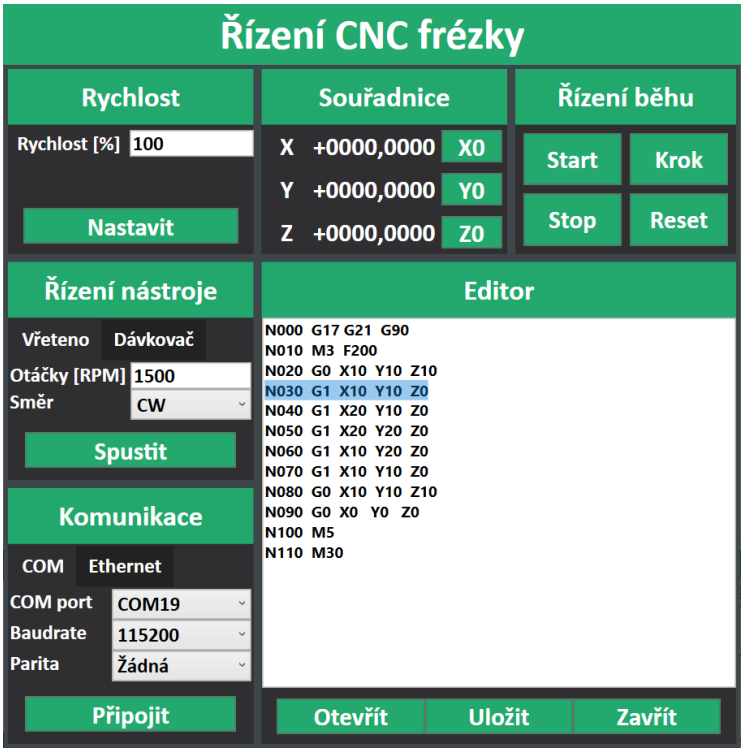
6 Realizace grafického rozhraní

Grafické rozhraní je implementováno dle návrhového vzoru MVVM - Model-View-ViewModel. Část model obsahuje komunikační jádro - CommunicationCore.cs a utilitu pro přístup k souborů - FileHelper.cs. Při startu aplikace se vytvoří instance těchto základních komponent a předají se do příslušných instancí ViewModel. Dále jsou implementovány třídy pro práci s rámci dat pro komunikační rozhraní. Protokol je popsán v kapitole Komunikační protokol.

ViewModel tvoří spojení mezi View a Modelem. Pro každé View je vytvořen jeden view model, který obsahuje Vlastnosti(Property) a Příkazy(Command) které jsou navázané na prvky ve View. Kařky ViewModel implementuje rozhraní INotifyPropertyChanged aby byla zajištěna propagace hodnot do View.

Vrstva View obsahuje jednotlivé ovládací prvky aplikace. Grafická část je rozdělena do několika menších částí (UserControl) které jsou vloženy do hlavního okna. Do každého View je vložen soubor *Styles.xaml*, který obsahuje jednotné styly pro všechny ovládací prvky. Rozměry všech prvků jsou definovány tak aby bylo dosaženo responzivního designu tj. přizpůsobení pro různé velikosti zobrazovací plochy.

Každý View má svůj datový kontext který je pomocí bindingu napojen na jednotlivé ViewModely.



Obr. 6.1: Grafický návrh

6.1 Editor

Umožnuje otevírání, editování a ukládání programu. Je navázán na ViewModel *EditorView.cs*. View je tvořeno prvkem RichTextBox, který zobrazuje aktuálně otevřený program. Ve spodní části jsou umístěny ovládací tlačítka Otevřít, Uložit a Zavřít. Událost kliknutí na jednotlivé tlačítka je navázána na události v nižší vrstvě. Grafický návrh je vidět na obrázku 6.1. Obsah prvku *RichTextBox* není možné navázat přímo na vlastnost třídy. V části *View* proto musela být přidána vlastnost *DependencyProperty*, pomocí které je možné vazbu vytvořit. Pro text v tomto prvku je možné nastavit různé styly. Zdrojem dat je soubor zvolený pomocí třídy *FileHelper*. Po kliknutí na tlačítko otevřít je zobrazen dialog pro zvolení zdrojového souboru. Následně je obsah načten do vlastnosti *EditorContent*. Prostý text je nutné převést na *FlowDocument*. Jednotlivé řádky souboru jsou vloženy do odstavců a jsou postupně přidány do obsahu editoru. Při ukládání je nutné udělat reverzní proces - převést obsah editoru zpět do prostého textu a uložit. Tato část je testována otevřením a uložením zdrojového souboru bez zásahu do editoru. Soubor nesmí být změněn pokud jsou oba převody provedeny správně.

6.2 Souřadnice

Tato část zobrazuje aktuální pozice jednotlivých os v milimetrech a umožňuje jejich nulování. Na pozadí běží cyklická úloha ve které se s využitím komunikačního jádra posílají požadavky *GET_POS* do procesoru. Odpověď na tento požadavek obsahuje 3 argumenty typu double v pořadí X, Y a Z. Dále jsou v pravé části 3 tlačítka pro nulování os. Po kliknutí se do procesoru odešle příkaz *ZERO* s parametrem názvu osy. Protokol je připraven i pro nulování všech os zároveň - příkaz *ZERO* s parametrem *ALL*. Tato funkcionalita je použitelná zatím jen při ovládání z konzole. Zobrazení čísel v grafickém rozhraní i v komunikačním protokolu bylo omezeno na 3 desetinná místa. Je to z důvodu zkrácení příkazu který je posílán mezi procesorem a počítačem.

6.3 Komunikace

V rámci ovládacího panelu je vytvořeno panelové zobrazení pro různé druhy komunikace. Výchozím typem je připojení pomocí COM portu. Zde lze nastavit příslušný port, jeho rychlost a paritu. Seznam dostupných portů je načten po startu aplikace. Druhý panel obsahuje nastavení pro připojení přes Ethernet. Zejména IP adresu a port. Toto připojení není v této verzi využíváno. Tento panel je možné rozšířit o další druhy komunikace.

6.4 Řízení běhu programu

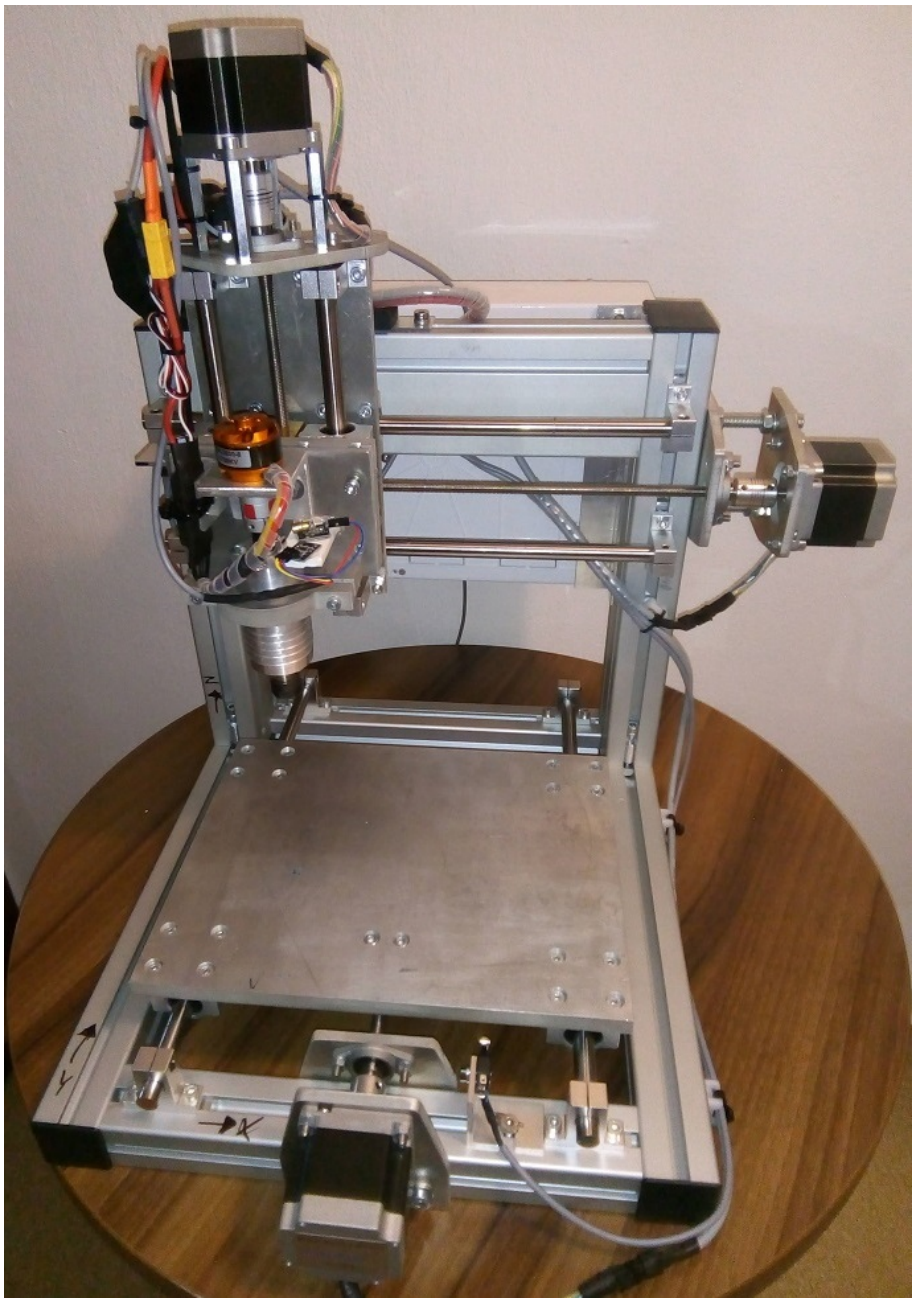
Tato část obsahuje 4 ovládací tlačítka: START, STOP, STEP a RESET. Jednotlivé události jsou navázané pomocí atributu *Command* do ViewModelu. Po vyvolání události (stisknutí tlačítka) dojde k vytvoření příslušného příkazu a odeslání do procesoru.

6.5 Testování

Pro testování grafického rozhraní byl použit vestavěný nástroj ve Visual Studiu. V rámci řešení je vytvořen projekt *CNC.Test* který obsahuje testy. Testují se hlavně operace spojené se zpracováním příkazu a jeho argumentů. Pro komunikační část jsou napsány testy pro ověření funkčnosti příkazu, které jsou závislé na připojeném HW. Automatické testování grafického rozhraní při tomto rozsahu aplikace zatím nebylo považováno za nutné.

7 Výsledky studentské práce

Na obrázku 7.1 je kompletní mechanická konstrukce stroje.



Obr. 7.1: Mechanická konstrukce stroje

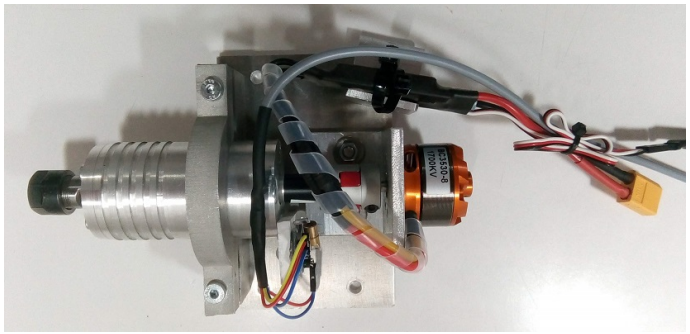
Rám z hliníkových profilů pro navrženou velikost stroje je tuhostí dostatečný pro měkké materiály. Kolmost řezů a délky dodaných profilů byly stěžejní pro dosažení kolmosti, která byla ověřena pomocí úhelníku a kontrolou shodnosti úhlopříček. Hlazené tyče tvořící lineární vedení byly ustaveny pomocí úchylkoměru dle profilů rámu7.2. Odchylka pro osu Y je nižší než rozlišení úchylkoměru na celé délce tj. pod

10 um. U osy X se projevuje deformace hlazených tyčí s maximem 80 um uprostřed rozsahu osy. Důvodem je zatížení osou Z s motorem a vřetenem.

Obr. 7.2: Ověření kolmosti pomocí úchylkoměru

Celý systém je řízen systémem na čipu Zynq osazeném na vývojovém kitu ZYBO. Pro krokové motory byly použity 4 ampérové budiče řízené pulzy. Pro motor vřetene je použit RC budič do 40 A. V systému jsou 3 napájecí napětí: 24 V pro krokové motory, 12 V pro vřeteno/ventil dávkovače a 5 V pro řídicí část. Silové části jsou odděleny od vývojového kitu optočlenů.

Pro stroj byly vyrobeny dva nástroje které je možné přišroubovat na přírubu osy Z. Při použití vřetene je nutné po instalaci nástroje ověřit kolmost osy kleštiny na osu X. Vřeteno včetně driveru a snímání otáček je možné vidět na obrázku 7.3.

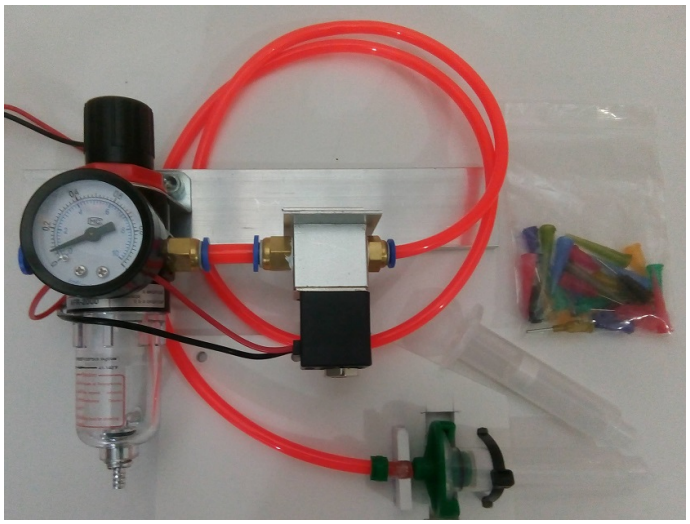


Obr. 7.3: Vřeteno

Do vřetene lze upnout nástroj s průměrem stopky 1 - 7 mm. Teoretická maximální rychlost je 30000 RPM. Z důvodu absence chlazení vřetene je rychlost omezena na 10000 RPM. U vřetene jsou vymezeny vůle a nedochází k vibracím.

Druhým nástrojem je dávkovač na obrázku 7.4. Na vstupu je regulátor tlaku s filtrem. Maximální vstupní tlak je 12 barů. Objem dávkovací stříkačky lze zvolit:

5, 10 nebo 30 centilitrů. Stříkačku lze osadit libovolnou jehlou. Primárně je dávkovač určen pro pájecí pastu a lepidlo.



Obr. 7.4: Dávkovač

Pro řízení krokového motoru byla vytvořena periferie s možností ovládání na sběrnici AXI. Výhodou je možnost řetězení na synchronizační signál - po spuštění první instance se synchronně spustí všechny další na stejném synchronizačním signálu. Lze vytvořit tří, šesti i více-osou aplikaci. Pro řízení vřetene jsou použity dva časovače od Xilinx: pro generování PWM a pro zpětnou vazbu.

Grafické rozhraní bylo vytvořeno dle teoretických předpokladů s důrazem na jednoduchost a intuitivnost. Vzhled je možné vidět v kapitole 6. Při používání nedochází k "zamrzání"programu díky použití asynchronních metod na pozadí. Zdrojový kód je přehledný díky použití standardního návrhového vzoru. Stroj je k počítači připojen pomocí USB portu. V systému se stroj zobrazí jako sériový port. Připojení je možné s parametry 115200/8N1. Zdrojový NC kód musí být vytvořen s využitím postprocesoru uvedeným v repozitáři projektu.

8 Závěr

V práci jsou popsány jednotlivé části CNC stroje. Dle teoretických poznatků, získaných z dostupných zdrojů a požadavků na stroj, byly v první části práce navrženy jednotlivé části systému. Systém byl následně realizován.

Návrh s procesorem ARM STM32 byl nahrazen návrhem na systému Zynq. Tento krok byl učiněn z důvodu škálovatelnosti celého systému pro více motorů, generování vyšších frekvencí a přesnější synchronizaci. Procesor není dostatečně rychlý pro jemnou interpolaci kruhových trajektorií, dále je návrh omezen počtem hardwarových časovačů a jejich funkcí. Konkrétně se jedná o řešení, ve kterém jsou data přijímána z virtuálního sériového portu, zpracovávána a předávána do AXI periferie obsahující řízení motorů. Veškeré knihovny byly psány v jazyce C++ a to tak, aby byly v budoucnu jednoduše přenositelné například pro řešení s RT Linuxovým jádrem. Veškeré zdrojové kódy jsou pod otevřenou licencí - je možné kteroukoliv část volně použít nebo přispět k vývoji. Zdrojové kódy jsou umístěny na serveru GitLab [28].

V projektu je kladen důraz na testování kódu v automatickém režimu. Jednotlivé části logiky jsou automaticky testovány na serveru

Bylo nastudováno používání AXI sběrnic pro komunikaci mezi procesorem a hradlovým polem. Pro sběrnici byla navržena periferie pro účely řízení krokového motoru. Jednotlivé VHDL popisy byly otestovány pomocí simulace. Výsledky simulací jsou uvedené v kapitole 5. Část pro úpravu délky výstupního pulzu byla nastavena dle použitého budiče.

Mechanická část byla navržena ve 2 verzích. Po konzultaci byl vyroben prototyp druhé verze z hliníkových profilů. Kolmost jednotlivých os byla ověřena pomocí úchylkoměru výsledky v jednotlivých osách jsou uvedeny v kapitole 7. Mezi hlavními požadavky byla víceúčelovost stroje. Z toho důvodu byla navržena příruba pro rychlou výměnu nástroje. Základním nástrojem je vřeteno pro frézování, které bylo navrženo a vyrobeno pro účely prototypu. Druhým nástrojem je dávkovač tvořený stříkačkou pro pájecí pastu. Dále je možné upnout například laser nebo vakuovou hlavu pro manipulaci s SMD součástkami.

Literatura

[1] *MillRight CNC Power Route Kit. In: Mill Right [online]. [cit. 2018-12-12].* Dostupné z: https://millrightcnc.com/product/millright-cnc-power-route-kit-bundle/

[2] *Číslicové řízení strojů. Terminologie os a pohybů. 2. vydání. Praha: Český normalizační institut, 2003.*

[3] *Building a Hobby CNC Router - Materials and Tools [online]. [cit. 2019-5-21].* Dostupné z: http://www.cncroutersource.com/hobby-cnc-router.html

[4] *Linear Rail CNC Machines | OpenBuilds. Build List | OpenBuilds [online]. Copyright © [cit. 13.05.2019].* Dostupné z: https://openbuilds.com/builds/linear-rail-cnc-machines.1771

[5] MAREK, Jiří. *Konstrukce CNC obráběcích strojů. Praha: MM publishing, 2006. ISBN 12122572.*

[6] *Předpětí ložisek. SKF [online]. [cit. 2019-5-21].* Dostupné z: http://www.skf.com/cz/products/bearings-units-housings/super-precision-bearings/principles/design-considerations/bearing_preload/index.html

[7] *Volba předpětí. SKF [online]. [cit. 2019-5-21].* http://www.skf.com/cz/products/bearings-units-housings/principles/bearing-selection-process/bearing-execution/internal-clearance-preload/selecting-preload/index.html

[8] Karlo Apro *Secrets of 5-Axis Machining. Industrial Press, Inc. 1st edition, 2008. ISBN 9780831133757.*

[9] *Rotary-Linear Motion Conversion Mechanism 4 [online]. [cit. 2019-5-21].* https://www.misumi-techcentral.com/tt/en/lca/2016/09/254-basic-elements-of-automation-clever-mechanisms-rotary-linearř-motion-con
html

[10] *Vnitřní vůle. SKF [online]. [cit. 2019-5-21].* https://www.skf.com/cz/products/bearings-units-housings/principles/general-bearing-knowledge/bearing-basics/internal-clearance/index.html

[11] *What Types of Motors Are Used on CNC Machines [online]. [cit. 2019-5-21].* Dostupné z: https://www.techwalla.com/articles/what-types-of-motors-are-used-on-cnc-machines

[12] *PID Theory Explained [online]. [cit. 2019-5-21].* Dostupné z: http://www.ni.com/en-my/innovations/white-papers/06/pid-theory-explained.html

[13] *Druhy krokových motorů. Katedra výpočetní a didaktické techniky [online]. 2017 [cit. 2019-5-21].* Dostupné z: https://www.kvd.zcu.cz/cz/materialy/POS/HTML/57/default.htm

[14] *Pružné spojky. T.E.A. TECHNIK [online]. [cit. 2019-5-21].* Dostupné z: https://www.teatechnik.cz/pruzne-spojky

[15] *K čemu slouží G kódy a M kódy. Macmatic [online]. 2016 [cit. 2019-5-21].* Dostupné z: https://www.macmatic.cz/component/content/article/40-technicke-clanky/66-k-cemu-slouzi-g-kody-a-m-kody

[16] *Smoothieboards. Smoothie [online]. 2017 [cit. 2019-5-21].* Dostupné z: http://smoothieware.org/smoothieboard

[17] *Zynq-7000. In: Digi-Key [online]. [cit. 2019-5-21].* Dostupné z: https://www.digikey.com/en/product-highlight/x/xilinx/zynq-7-soc

[18] *Onshape Education. Onshape [online]. 2014 [cit. 2019-5-21].* Dostupné z: https://www.onshape.com/products/education

[19] *Speed Limits [cit. 21.05.2019].* Dostupné z: https://www.amroll.com/speed-limits.html

[20] *Real-Time Linux. Xilinx Wiki [online]. 2018 [cit. 2019-5-21].* Dostupné z: https://xilinx-wiki.atlassian.net/wiki/spaces/A/pages/18842435/Real-Time+Linux

[21] *Ziegler-Nichols Method. Michigan Technological University [online]. 2004 [cit. 2019-5-21].* Dostupné z: http://pages.mtu.edu/~tbco/cm416/zn

[22] *Guide to editing a Vectric Post Processor [cit. 21.05.2019].* Dostupné z: https://altlab.org/d/s/projectos/roland-mod-ela/aspire_post_processor_guide.pdf

[23] *BIES, Lammert. LibCRC. LibCRC – Open Source CRC Library in C [online]. 2016 [cit. 2019-5-21].* Dostupné z: https://www.libcrc.org

[24] *Zybo Zynq-7000 ARM/FPGA SoC Trainer Board. Digilent Inc [online]. 2016 [cit. 2019-5-21].* Dostupné z: https://store.digilentinc.com/zybo-zynq-7000-arm-fpga-soc-trainer-board/

[25] *CppUTest unit testing and mocking framework for C/C++ [cit. 2019-5-21].* Dostupné z: https://cpputest.github.io

[26] *AXI Timer v2.0, LogiCORE IP Product Guide [cit. 2019-5-21].* Dostupné z: https://www.xilinx.com/support/documentation/ip_documentation/axi_timer/v2_0/pg079-axi-timer.pdf

[27] *ZYBO™ FPGA Board Reference Manual [cit. 2019-5-21].* Dostupné z: https://reference.digilentinc.com/_media/reference/programmable-logic/zybo/zybo_rm.pdf

[28] *GitLab projekt soc-cnc [online]. 2018 [cit. 2019-5-21].* Dostupné z: https://gitlab.com/danielmichalik/soc-cnc

Seznam obrázků

Seznam symbolů, veličin a zkratek

CNC	Computer Numeric Control - číslicové řízení s využitím počítače
SMD	Surface mount device - součástka pro povrchovou montáž
SoC	System on chip - systm na čipu
AXI	Advanced Extensible Interface
PS	Processing system - část s procesorem
PL	Programmable logic - programovatelná logika
USB	Universal Serial Bus - univerzální sériová sběrnice
I2C	Inter-Integrated Circuit - sériová sběrnice
SPI	Serial peripheral interface - sériové periferní rozhraní
UART	Universal asynchronous receiver and transmitter - univerzální asynchronní sériové rozhraní
CAN	Controlled Area Network - datová sběrnice
SD	Secure Digital - paměťová karta
PID	Proporcionálně integračně derivační regulátor
CAM	Computer Aided Manufacturing - Počítačem podporovaná výroba
STX	Start of text - začátek textu
ETX	End of text - konec textu
ot./V	Otáček na volt
RT	Realtime
RPM	Revolutions per minute - otáčky za minutu
FPU	Floating-point unit - koprocesor pro práci s typem float
PMOD	Peripheral Module interface - modulární rozhraní periferie